

# PROJECT REPORT

---

**Course Code:** CSEG1032

**Course Title:** Programming in C

**Project Title:** Contact Management System

**Student Name:** Prince Raj Rawat

**SAP ID:** 590022507

**Semester:** 1<sup>st</sup>

---

## 1. ABSTRACT

This project implements a menu-driven Contact Management System using the C programming language. The program allows the user to add, view, search, and delete contact records stored persistently in a file. It demonstrates the use of structures, file handling, and modular programming techniques to build a scalable and maintainable software system.

## 2. OBJECTIVE

The objective of this project is to apply fundamental and intermediate concepts of the C language to design a working application. It aims to help store contact information efficiently.

- Use structures to represent complex data.
- Employ functions and modular design for code organization.
- Perform file operations for persistent data storage.

### **3. PROBLEM DEFINITION**

Almost every people require simple tools to maintain contact . Manually handling these records is error-prone and inefficient. This project provides a basic command-line system for managing contact information such as Name, Phone Number, and Email Address in a text file.

### **4. SYSTEM DESIGN AND ALGORITHM**

The system is modular, consisting of multiple source files:

- main.c controls program flow and user interaction.
- call.c handles CRUD (Create, Read, Update/Delete) operations.

### **5. IMPLEMENTATION DETAILS**

Key Language Features Used:

- Structures to represent contact info.
- File Handling functions (fopen, fwrite, fread, fclose) to store and retrieve data.
- Functions for modularity and clarity.

## Example Code Snippet

### 6. OUTPUT (Sample)

Menu:

```
1.1.Add Contact
  2. View Contacts
  3. Search Contact
  4. Delete Contact
  5. Exit
2.INPUT : 1
```

### Sample Output

```
Enter Name: John Doe
Enter Phone: 9876543210
Enter Email: john@example.com
Contact added successfully!
```

### 7. CONCLUSION

The Contact Management System demonstrates the practical application of C programming concepts such as structures, file handling, and modular design. It provides a simple yet effective solution for managing contact records in small organizations.

### 8. FUTURE ENHANCEMENTS

- Add Update Contact feature.
- add grouping in contact like family, friends , classmate etc.
- Create a GUI-based interface for better usability.

### 9. APPENDIX:

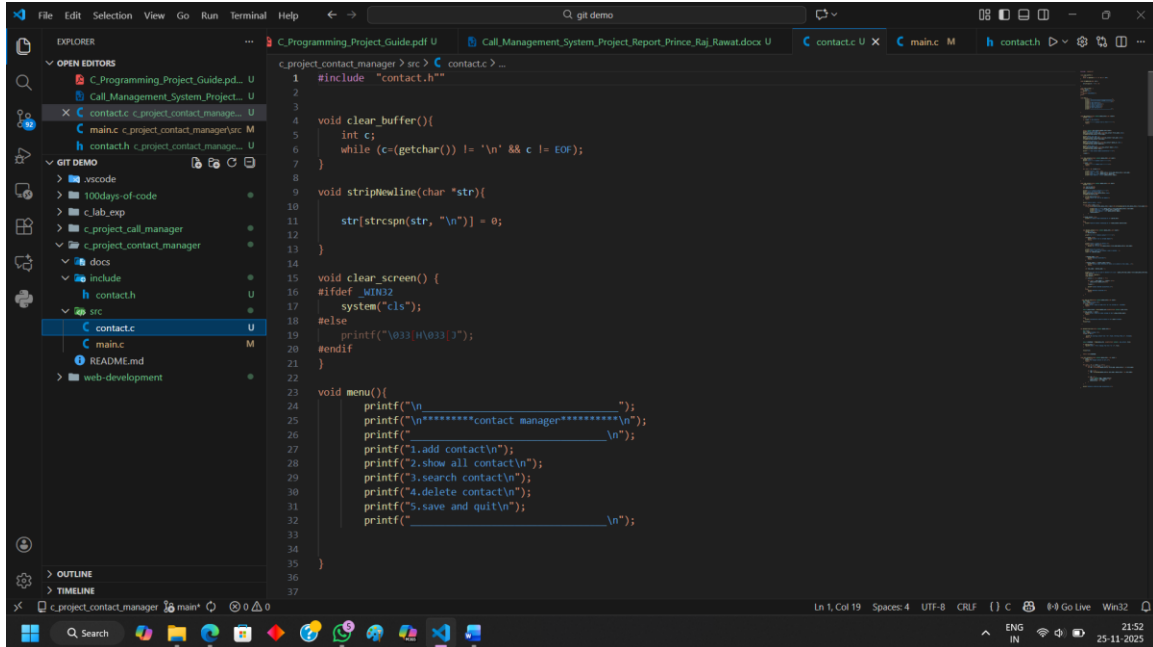
## main.c (Provided)

```
c_project_contact_manager > src > main.c > ...
1 { #include "contact.h"
2
3
4 int main(){
5
6     struct contact phone_info[max_contact];
7     int count=0;
8     int opt=0;
9
10    printf("starting contact manager..\n");
11
12    count = upload_from_file(phone_info);
13    printf("Loaded %d contacts from file.\n", count);
14
15
16
17    while(opt != 5)
18    {
19        menu();
20
21        printf("\nenter :");
22        scanf("%d",&opt);
23        clear_buffer();
24
25        switch(opt){
26
27            case 1 :
28                {clear_screen();
29                 add_contact(phone_info, &count);
30                 sort_contacts(phone_info, count);
31                 break;}
32            case 2 :
33                {clear_screen();
34                 show_contacts(phone_info, count);
35                 break;}
36            case 3 :
37                {clear_screen();
```

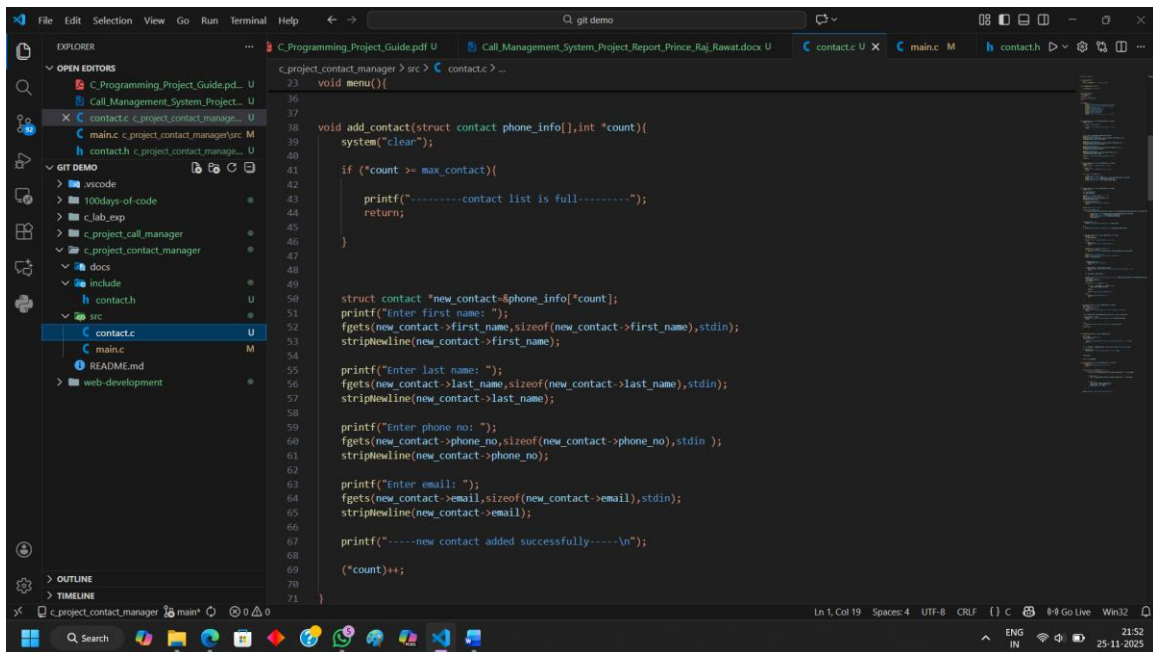
```

38
39                break;}
40            case 4 :
41                {clear_screen();
42                 delete_contact(phone_info, &count);
43                 break;}
44            case 5 :
45                {save_in_file(phone_info, count);
46                 break;}
47            default:
48                {printf("invalid choice please choose no. from 1-5.\n");}
49        }
50
51
52
53    }
54
55
56
57    return 0;
58
59 }
60
```

# contact.c



```
1 #include "contact.h"
2
3
4 void clear_buffer(){
5     int c;
6     while (c=(getchar()) != '\n' && c != EOF);
7 }
8
9 void stripNewline(char *str){
10
11     str[strcspn(str, "\n")] = 0;
12 }
13
14
15 void clear_screen() {
16     #ifdef WIN32
17         system("cls");
18     #else
19         printf("\033[H\033[J");
20     #endif
21 }
22
23 void menu(){
24     printf("\n");
25     printf("\n*****contact manager*****\n");
26     printf("\n");
27     printf("1.add contact\n");
28     printf("2.show all contact\n");
29     printf("3.search contact\n");
30     printf("4.delete contact\n");
31     printf("5.save and quit\n");
32     printf("\n");
33 }
34
35
36
37
```



```
38 void menu(){
39
40 }
41
42 void add_contact(struct contact phone_info[],int *count){
43     system("clear");
44
45     if (*count >= max_contact){
46         printf("-----contact list is full-----");
47         return;
48     }
49
50     struct contact *new_contact=&phone_info[*count];
51     printf("Enter first name: ");
52     fgets(new_contact->first_name,sizeof(new_contact->first_name),stdin);
53     stripNewline(new_contact->first_name);
54
55     printf("Enter last name: ");
56     fgets(new_contact->last_name,sizeof(new_contact->last_name),stdin);
57     stripNewline(new_contact->last_name);
58
59     printf("Enter phone no: ");
60     fgets(new_contact->phone_no,sizeof(new_contact->phone_no),stdin );
61     stripNewline(new_contact->phone_no);
62
63     printf("Enter email: ");
64     fgets(new_contact->email,sizeof(new_contact->email),stdin);
65     stripNewline(new_contact->email);
66
67     printf("-----new contact added successfully-----\n");
68
69     (*count)++;
70 }
71
72
```

This screenshot shows the VS Code editor with the `contact.c` file open. The Explorer sidebar on the left shows the project structure, including `src`, `include`, `contact.h`, `main.c`, `README.md`, and `web-development`. The main editor displays the following C code:

```
38 void add_contact(struct contact phone_info[],int *count){
39     }
40 }
41
42 void show_contacts(struct contact phone_info[],int count){
43     system("clear");
44     printf("\n\n-----contact list-----\n\n");
45
46     if(count == 0){
47         printf("-----empty list-----\n\n");
48         return;
49     }
50
51     for (int i = 0; i<count;i++){
52
53         printf("contact : %d\n", i+1);
54         printf("name : %s %s\n", phone_info[i].first_name,phone_info[i].last_name);
55         printf("phone no : %s\n",phone_info[i].phone_no);
56         printf("email address : %s\n\n",phone_info[i].email);
57     }
58 }
59
60 void find_contact(struct contact phone_info[],int count){
61     system("clear");
62
63     char search_term[50];
64     int found_contact=0;
65
66     printf("-----search contact-----\n\n");
67     printf("enter the search term: ");
68     fgets(search_term,sizeof(search_term),stdin);
69     stripnewline(search_term);
70
71     if (strlen(search_term)==0){
72         printf("search term can not be empty\n");
73     }
```

This screenshot shows the VS Code editor with the `contact.c` file open, displaying the continuation of the C code from the previous image:

```
74 void find_contact(struct contact phone_info[],int count){
75     stripnewline(search_term);
76
77     if (strlen(search_term)==0){
78         printf("search term can not be empty\n");
79         return;
80     }
81
82     printf("search result...\n\n");
83
84     for(int i=0;i< count; i++){
85         if(strcasecmp(search_term,phone_info[i].email)==0||strcasecmp(search_term,phone_info[i].first_name)==0||strcasecmp(search_t
86             {
87                 printf("name: %s %s\n",phone_info[i].first_name,phone_info[i].last_name);
88                 printf("phone no. : %s\n",phone_info[i].phone_no);
89                 printf("email address : %s\n",phone_info[i].email);
90                 found_contact++;
91             }
92         }
93
94     if(found_contact == 0){
95         printf("no contact found containing %s \n",search_term);
96     }
97     else{
98         printf("found %d contact containing %s \n",found_contact,search_term);
99     }
100 }
101
102 void delete_contact(struct contact phone_info[],int *count){
103     system("clear");
104     int delete_index=0;
105
106     printf("\n\n-----delete contact-----\n\n");
```

This screenshot shows a Visual Studio Code editor window with the file explorer on the left and the source code in the center. The file explorer shows a project structure with folders like 'src', 'include', and 'docs'. The 'src' folder is expanded, showing 'contact.c' as the active file. The code in the center is the 'delete\_contact' function, which takes a pointer to a 'contact' array and its size as arguments. It starts by clearing the screen and initializing a 'delete\_index' variable. It then prints a message to delete a contact. A loop iterates through the 'contact' array, printing each contact's details. After the loop, it prompts the user to press 0 to cancel or select an index to delete. It then checks if the selected index is valid and if there are contacts to delete. If valid, it decrements the 'real\_index' and prints the contact to be deleted. The function ends by printing a confirmation message.

```
132 void delete_contact(struct contact phone_info[],int *count){
133     system("clear");
134     int delete_index=0;
135
136     printf("\n-----delete contact-----\n");
137
138     if(*count == 0){
139         printf("contact list is already empty\n");
140         return;
141     }
142
143     printf("select a contact to delete.\n");
144     for(int i=0;i<*count;i++){
145         printf("%d. %s %s\n",i+1,phone_info[i].first_name,phone_info[i].last_name);
146     }
147
148     printf("press 0 to cancel.\n");
149     printf("select index of the contact u want to delete : ");
150     scanf("%d",&delete_index);
151
152
153
154
155     if(delete_index == 0){
156         printf("deletion cancelled\n");
157         return;
158     }
159
160     if(delete_index<1 || delete_index>*count){
161         printf("Invalid index selection,\n there is no contact on this index...\n");
162         return;
163     }
164
165     int real_index = delete_index - 1;
166
167     printf("are you sure you want to delete %s %s (y/n):",phone_info[real_index].first_name,phone_info[real_index].last_name);
```

This screenshot shows the same Visual Studio Code editor window, but now displaying the 'save\_in\_file' function. This function takes a pointer to a 'contact' array and its size as arguments. It opens a file for writing in binary mode. If the file cannot be opened, it prints an error message. It then writes the entire 'contact' array to the file. After writing, it prints the number of items written. If the number of items written is not equal to the count, it prints an error message. The function ends by returning.

```
168
169
170 void save_in_file(struct contact phone_info[],int count){
171     FILE *file;
172     file = fopen(filename,"wb");
173     if (file == NULL){
174         printf("Error: Could not open file '%s' for writing.\n", filename);
175         return;
176     }
177
178     size_t item_written = fwrite(phone_info,sizeof(struct contact),count,file);
179
180     if(item_written != count){
181         printf("Error: write %zu items instead of %d\n",item_written,count);
182         return;
183     }
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
---
```

```
void save_in_file(struct contact phone_info[], int count){
    if(item_written != count){
        else{
            printf("successfully saved %d contact to %s",count,filename);
        }
        fclose(file);
    }
}

int upload_from_file(struct contact phone_info[]){
    FILE *file;
    file = fopen(filename,"rb");
    if(file == NULL){
        printf("No existing contact file ('%s') found. Starting fresh.\n", filename);
        return 0;
    }

    size_t itemsRead = fread(phone_info, sizeof(struct contact), max_contact, file);

    if (ferror(file)) {
        fprintf(stderr, "Error reading from file '%s'.\n", file);
    }

    fclose(file);

    return (int)itemsRead;
}

void sort_contacts(struct contact phone_info[], int count) {
```

```
int upload_from_file(struct contact phone_info[]){
}

void sort_contacts(struct contact phone_info[], int count) {
    if (count < 2) {
        printf("Not enough contacts to sort.\n");
        return;
    }

    for (int i = 0; i < count - 1; i++) {
        for (int j = 0; j < count - i - 1; j++) {
            int cmp = strcasecmp(phone_info[j].first_name, phone_info[j + 1].first_name);

            if (cmp == 0) {
                cmp = strcasecmp(phone_info[j].last_name, phone_info[j + 1].last_name);
            }

            if (cmp > 0) {
                struct contact temp = phone_info[j];
                phone_info[j] = phone_info[j + 1];
                phone_info[j + 1] = temp;
            }
        }
    }

    printf("Contacts sorted by name successfully.\n");
}
```