DESIGN AND ANALYSIS OF ALGORITHM

ALGORITHM ANALYSIS

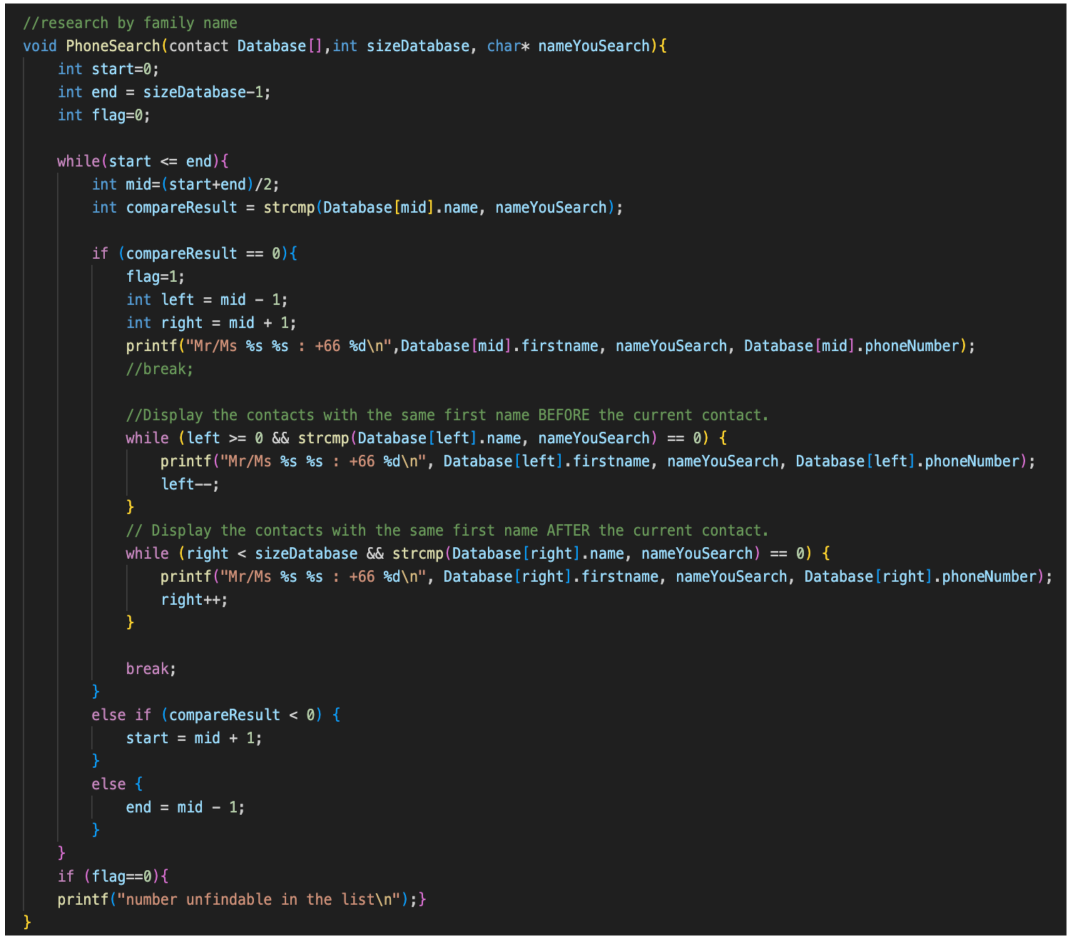
Submitted by
Corentin GIGOT (ES04084)
Morgat COCHENNEC (ES04094)
Zoë SAVI (ES04034)
Léa FOUGERA-LEMPEREUR (ES04032)
Group name: First call

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

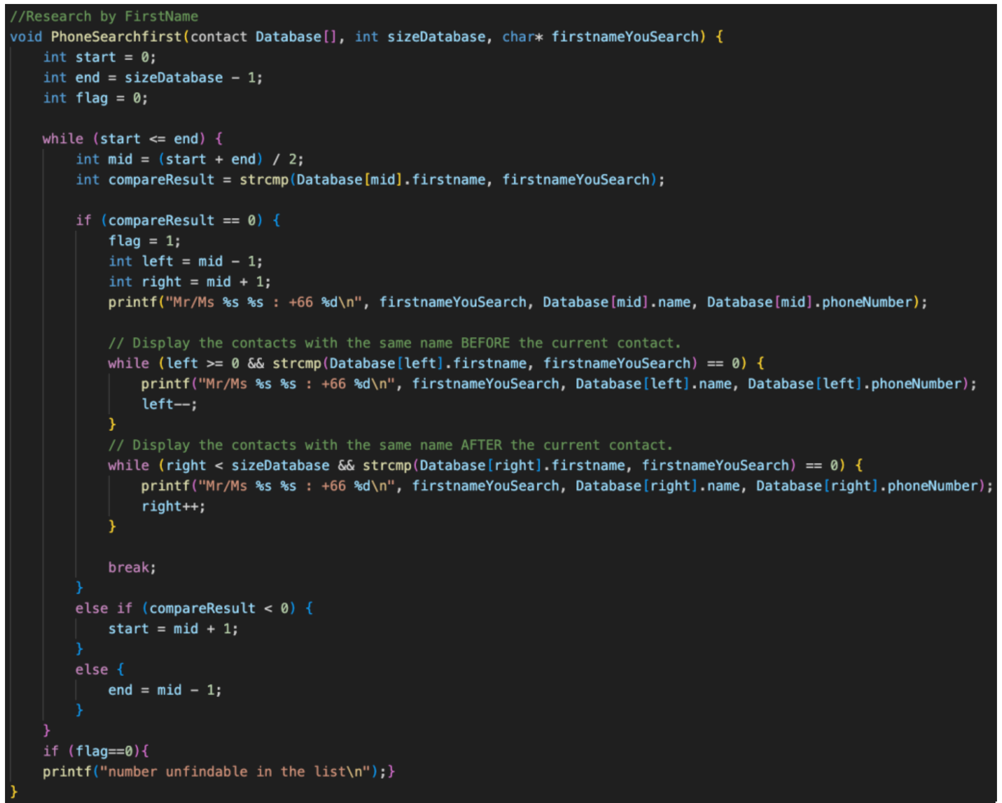
SEM 2 2022-2023

With the help of the PhoneSearch function, you may look for a name in a contact database and see the matching phone numbers. In order to expedite the search, it employs a binary search strategy.
An array of contacts (Database), its size (sizeDatabase), and the name you're looking for (nameYouSearch) are all inputs to the function. The search range is defined by the start and end indices.
By contrasting the sought name with the name of the contact in the centre of the range using a while loop, the function does a binary search. The function modifies the start and end indices to continue the search in the appropriate half of the range depending on the outcome of the comparison.
If the name is recognised, the function shows the contact's first name, last name, and phone number. The database is then looped through to display all previous and subsequent contacts with the same name as the current contact.
The function shows a message stating that the number cannot be found in the list if the name is not found.
This tool quickly locates phone numbers connected to a given name in a database of contacts ordered by name using binary search.



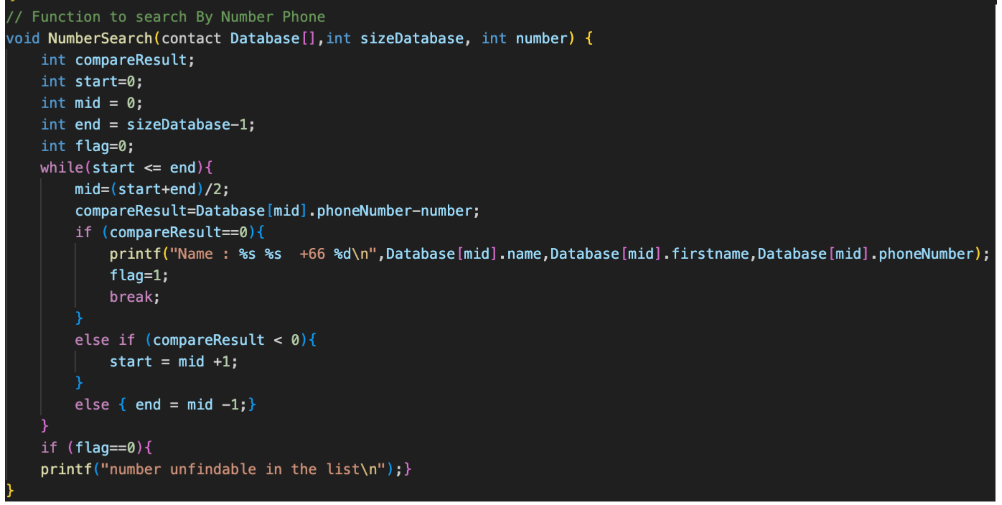
```c
//research by family name
void PhoneSearch(contact Database[],int sizeDatabase, char* nameYouSearch){
    int start=0;
    int end = sizeDatabase-1;
    int flag=0;

    while(start <= end){
        int mid=(start+end)/2;
        int compareResult = strcmp(Database[mid].name, nameYouSearch);

        if (compareResult == 0){
            flag=1;
            int left = mid - 1;
            int right = mid + 1;
            printf("Mr/Ms %s %s : +66 %d\n",Database[mid].firstname, nameYouSearch, Database[mid].phoneNumber);
            //break;

            //Display the contacts with the same first name BEFORE the current contact.
            while (left >= 0 && strcmp(Database[left].name, nameYouSearch) == 0) {
                printf("Mr/Ms %s %s : +66 %d\n", Database[left].firstname, nameYouSearch, Database[left].phoneNumber);
                left--;
            }
            // Display the contacts with the same first name AFTER the current contact.
            while (right < sizeDatabase && strcmp(Database[right].name, nameYouSearch) == 0) {
                printf("Mr/Ms %s %s : +66 %d\n", Database[right].firstname, nameYouSearch, Database[right].phoneNumber);
                right++;
            }

            break;
        }
        else if (compareResult < 0) {
            start = mid + 1;
        }
        else {
            end = mid - 1;
        }
    }
    if (flag==0){
    printf("number unfindable in the list\n");}
}
```

We use the same model for the function to search by first name



```c
//Research by FirstName
void PhoneSearchfirst(contact Database[], int sizeDatabase, char* firstnameYouSearch) {
    int start = 0;
    int end = sizeDatabase - 1;
    int flag = 0;

    while (start <= end) {
        int mid = (start + end) / 2;
        int compareResult = strcmp(Database[mid].firstname, firstnameYouSearch);

        if (compareResult == 0) {
            flag = 1;
            int left = mid - 1;
            int right = mid + 1;
            printf("Mr/Ms %s %s : +66 %d\n", firstnameYouSearch, Database[mid].name, Database[mid].phoneNumber);

            // Display the contacts with the same name BEFORE the current contact.
            while (left >= 0 && strcmp(Database[left].firstname, firstnameYouSearch) == 0) {
                printf("Mr/Ms %s %s : +66 %d\n", firstnameYouSearch, Database[left].name, Database[left].phoneNumber);
                left--;
            }
            // Display the contacts with the same name AFTER the current contact.
            while (right < sizeDatabase && strcmp(Database[right].firstname, firstnameYouSearch) == 0) {
                printf("Mr/Ms %s %s : +66 %d\n", firstnameYouSearch, Database[right].name, Database[right].phoneNumber);
                right++;
            }

            break;
        }
        else if (compareResult < 0) {
            start = mid + 1;
        }
        else {
            end = mid - 1;
        }
    }
    if (flag==0){
    printf("number unfindable in the list\n");}
}
```

We use the same model for the function to search by phone number



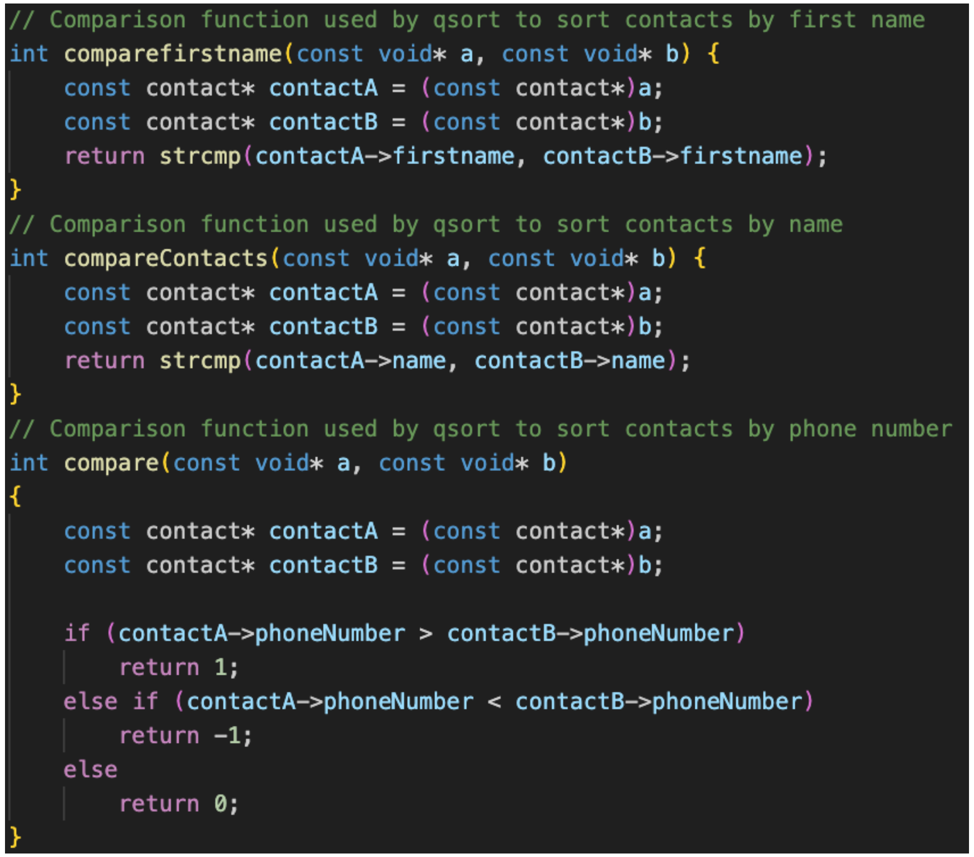
```c
// Function to search By Number Phone
void NumberSearch(contact Database[],int sizeDatabase, int number) {
    int compareResult;
    int start=0;
    int mid = 0;
    int end = sizeDatabase-1;
    int flag=0;
    while(start <= end){
        mid=(start+end)/2;
        compareResult=Database[mid].phoneNumber-number;
        if (compareResult==0){
            printf("Name : %s %s  +66 %d\n",Database[mid].name,Database[mid].firstname,Database[mid].phoneNumber);
            flag=1;
            break;
        }
        else if (compareResult < 0){
            start = mid +1;
        }
        else { end = mid -1;}
    }
    if (flag==0){
    printf("number unfindable in the list\n");}
}
```

We have several comparison functions that use the quicksort. These functions are all based on the same principle. They are then used in the search functions by name, by first name and by telephone number.

- I detail here the comparison function to sort contacts by name.

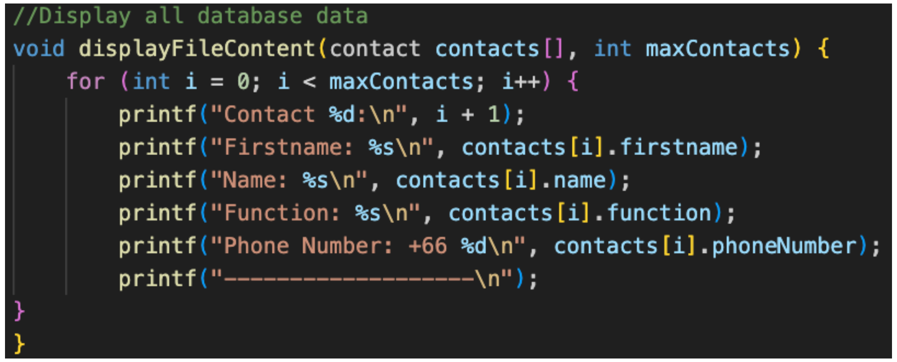
  Parameters ‘a’ and ‘b’ are pointers that point to the contact structure, which is in main.h, where people's information is stored in the database. They are then used for the comparison of character strings.
  The strcmp function compares the two strings and returns an integer indicating their sorting relationship. If the first names are the same, it returns 0. If contactA's first name is considered "before" contactB's first name, it returns a negative integer. Otherwise, it returns a positive integer.
- For the function that searches by first name, it's the same thing except that we use first names.
- For the function that searches by phone number, the principle of pointers is the same as for the other two functions.

  The numbers are compared. If contactA's phone number is less than contactB's phone number, the function returns -1 to indicate that contactA should be "before" contactB in the sort. If contactA and contactB's phone numbers are the same, the function returns 0, which means that both contacts are considered equal in terms of sorting.



```c
// Comparison function used by qsort to sort contacts by first name
int comparefirstname(const void* a, const void* b) {
    const contact* contactA = (const contact*)a;
    const contact* contactB = (const contact*)b;
    return strcmp(contactA->firstname, contactB->firstname);
}
// Comparison function used by qsort to sort contacts by name
int compareContacts(const void* a, const void* b) {
    const contact* contactA = (const contact*)a;
    const contact* contactB = (const contact*)b;
    return strcmp(contactA->name, contactB->name);
}
// Comparison function used by qsort to sort contacts by phone number
int compare(const void* a, const void* b)
{
    const contact* contactA = (const contact*)a;
    const contact* contactB = (const contact*)b;

    if (contactA->phoneNumber > contactB->phoneNumber)
        return 1;
    else if (contactA->phoneNumber < contactB->phoneNumber)
        return -1;
    else
        return 0;
}
```

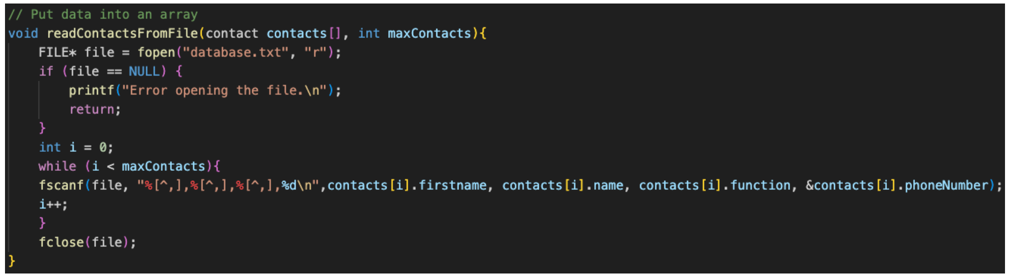
The function takes two input parameters: contacts[], which is an array of contacts, and maxContacts, which is the maximum number of contacts in the array.
A for loop is used to loop through each contact in the array. The variable i is used as a counter, starting at 0 and ending when i reaches maxContacts.
Inside the loop, printf statements are used to display each contact's information.
The following printf statements display the specific details of each contact.



```c
//Display all database data
void displayFileContent(contact contacts[], int maxContacts) {
    for (int i = 0; i < maxContacts; i++) {
        printf("Contact %d:\n", i + 1);
        printf("Firstname: %s\n", contacts[i].firstname);
        printf("Name: %s\n", contacts[i].name);
        printf("Function: %s\n", contacts[i].function);
        printf("Phone Number: +66 %d\n", contacts[i].phoneNumber);
        printf("-------------------\n");
}
}
```

This function takes two input parameters: contacts[], which is an array of contacts, and maxContacts, which is the maximum number of contacts the array can contain.

The function opens the "database.txt" file using the fopen function. If opening the file fails, the function displays an error message and exits with a return.
A counter i is initialized to 0 to browse the table of contacts.
The while loop is used to read data from the file as long as the counter i is less than maxContacts.
Inside the loop, the fscanf function is used to read contact information from the file. A format is specified in fscanf which means that the function will read a string up to the first comma and so on. The '\n' at the end indicates the end of the line in the file.
The values read from the file are stored in the corresponding places of the structure.
Counter i is incremented to go to the next contact in the table.
After all contacts have been read and stored, the function closes the file using fclose.



```c
// Put data into an array
void readContactsFromFile(contact contacts[], int maxContacts){
    FILE* file = fopen("database.txt", "r");
    if (file == NULL) {
        printf("Error opening the file.\n");
        return;
    }
    int i = 0;
    while (i < maxContacts){
    fscanf(file, "%[^,],%[^,],%[^,],%d\n",contacts[i].firstname, contacts[i].name, contacts[i].function, &contacts[i].phoneNumber);
    i++;
    }
    fclose(file);
}
```

The function takes two input parameters: Database[], which is an array of contacts, and sizeDatabase, which is the size of the database.

The function displays a menu allowing the user to choose the role of the person sought. The various options are displayed using the printf function.

The user is prompted to enter their choice using the scanf function, and the value is stored in the choice variable.

The function then uses a switch structure to perform different actions based on the user's choice.

For each case, the function performs a for loop to iterate through the contact database.

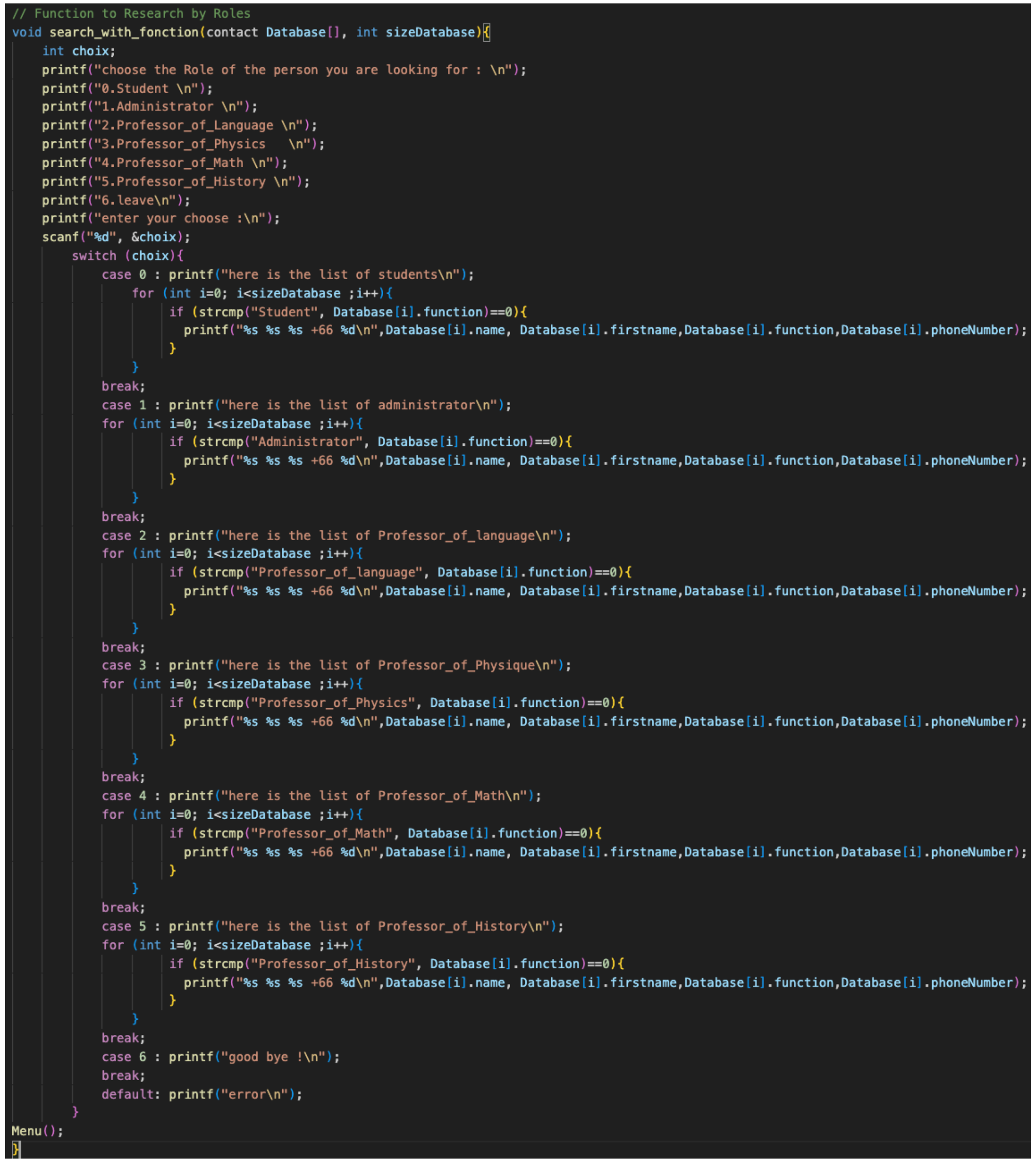
Inside the loop, the function checks if the function of the current contact matches the selected role using the strcmp function. If this is the case, the contact information (surname, first name, function, telephone number) is displayed on the screen using the printf function.

After each case, there is a break to exit the switch structure and continue executing the rest of the function.

If the user chooses option 6, the function displays "good bye!" and ends.

If the user enters an unrecognized value, the function displays "error".

After the switch structure, the function calls another Menu() function. This function displays the larger menu where the user can choose other options.



```c
// Function to Research by Roles
void search_with_fonction(contact Database[], int sizeDatabase){
    int choix;
    printf("choose the Role of the person you are looking for : \n");
    printf("0.Student \n");
    printf("1.Administrator \n");
    printf("2.Professor_of_Language \n");
    printf("3.Professor_of_Physics   \n");
    printf("4.Professor_of_Math \n");
    printf("5.Professor_of_History \n");
    printf("6.leave\n");
    printf("enter your choose :\n");
    scanf("%d", &choix);
        switch (choix){
            case 0 : printf("here is the list of students\n");
                for (int i=0; i<sizeDatabase ;i++){
                     if (strcmp("Student", Database[i].function)==0){
                       printf("%s %s %s +66 %d\n",Database[i].name, Database[i].firstname,Database[i].function,Database[i].phoneNumber);
                     }
                }
            break;
            case 1 : printf("here is the list of administrator\n");
            for (int i=0; i<sizeDatabase ;i++){
                     if (strcmp("Administrator", Database[i].function)==0){
                       printf("%s %s %s +66 %d\n",Database[i].name, Database[i].firstname,Database[i].function,Database[i].phoneNumber);
                     }
                }
            break;
            case 2 : printf("here is the list of Professor_of_language\n");
            for (int i=0; i<sizeDatabase ;i++){
                     if (strcmp("Professor_of_language", Database[i].function)==0){
                       printf("%s %s %s +66 %d\n",Database[i].name, Database[i].firstname,Database[i].function,Database[i].phoneNumber);
                     }
                }
            break;
            case 3 : printf("here is the list of Professor_of_Physique\n");
            for (int i=0; i<sizeDatabase ;i++){
                     if (strcmp("Professor_of_Physics", Database[i].function)==0){
                       printf("%s %s %s +66 %d\n",Database[i].name, Database[i].firstname,Database[i].function,Database[i].phoneNumber);
                     }
                }
            break;
            case 4 : printf("here is the list of Professor_of_Math\n");
            for (int i=0; i<sizeDatabase ;i++){
                     if (strcmp("Professor_of_Math", Database[i].function)==0){
                       printf("%s %s %s +66 %d\n",Database[i].name, Database[i].firstname,Database[i].function,Database[i].phoneNumber);
                     }
                }
            break;
            case 5 : printf("here is the list of Professor_of_History\n");
            for (int i=0; i<sizeDatabase ;i++){
                     if (strcmp("Professor_of_History", Database[i].function)==0){
                       printf("%s %s %s +66 %d\n",Database[i].name, Database[i].firstname,Database[i].function,Database[i].phoneNumber);
                     }
                }
            break;
            case 6 : printf("good bye !\n");
            break;
            default: printf("error\n");
        }
Menu();
}
```