Contact Management System

Project Report

In contact management system (CMS) we are instructed to design and implement a system which will allow use to add, delete and search contact from contacts list. A contact may contain name of user, phone number and date of birth. At most, 100 people’s contact

information can be stored. A name can be at most 20bytes excluding ‘\0’, a phone

number can be at most 15byes without ‘-’ (numbers only), a birth date is stored in a

YYYYMMDD format using 8 bytes. No spaces, tabs, new lines are allowed. Do not

evaluate the contact information. (One string received by scanf). No two persons have

the same name, but other information can be identical.

Use a structure to store one person’s information (name, phone number, birth date).

Use a structure array to store multiple people’s information. If necessary, allow to store

additional information in a structure. Names are case-sensitive

Major functionalities of CMS are:

1. Add a new contact into system
2. Delete a contact via username
3. Search a contact via month of birth of user/users.
4. Display all contact in list

“Registration” is to receive contact information (name, phone number, birth date) from a user and store them in the program. “Delete” is find and delete one’s contact information (name, phone number, birth date). “FindByBirth” is to display the people’s contact information who were born in a particular month. “ShowAll” is to display all the contact information (name, phone number, birth date) that is stored in the program.

Stage – 1

Objective of stage – 1 is to implement system’s major functionality using simple arrays.

**Structure of contact:**

|  |
| --- |
| struct Contact {  char name[21];  char phoneNumber[16];  char birthdate[9];  }; |

1. **Create a new contact:**

|  |
| --- |
| int registration(struct Contact contacts[100], int count){  printf("Name:");  scanf("%s", contacts[count].name);  printf("Phone\_number:");  scanf("%s", contacts[count].phoneNumber);  printf("Birth:");  scanf("%s", contacts[count].birthdate);  int i, j, tempIndex;  struct Contact temp;  for(i=0; i<count; i++) {  tempIndex = i;  for(j=i+1; j<=count; j++){  if(strcmp(contacts[j].name, contacts[tempIndex].name) < 0)  tempIndex = j;  }  temp = contacts[tempIndex];  contacts[tempIndex] = contacts[i];  contacts[i] = temp;  }  printf("<<%d>>\n", ++count);  return count;  } |

Parameters:

1. Array of contacts
2. Count of total contacts in array

Return:

It will return new count after adding a new contact

Function will take inputs from users which will contain username, phone number and birthdate. Then it will store this data into its cross-ponding index of array. Moreover, it will also sort array after each insertion.

1. **Delete a contact:**

|  |
| --- |
| int delete(struct Contact contacts[100], int count){  if(count == 0){  printf("NO MEMBER\n");  return count;  }  char name[21];  printf("Name:");  scanf("%s", name);  int i;  for(i=0; i<count; i++){  if(strcmp(contacts[i].name, name) == 0){  int j;  for(j=i; j<count-1; j++)  contacts[j] = contacts[j+1];  --count;  }  }  return count;  } |

Parameters:

1. Array of contacts
2. Count of total contacts in array

Return:

It will return new count after adding a new contact

Function will take username from users which will be use to search a contact with that name and then delete it. Delete a contact is done by overwriting a contact with next contact of it.

1. **Display all contacts:**

|  |
| --- |
| void showAll(struct Contact contacts[100], int count){  int i;  for(i=0; i<count; i++)  printf("%s %s %s\n", contacts[i].name, contacts[i].phoneNumber, contacts[i].birthdate);  } |

Parameters:

1. Array of contacts
2. Count of total contacts in array

Return:

void

This function will display all contacts present in an array; order of printing will be same as they are inserted. display all the contact information stored in the program by the

order that they are stored. Display one person’s information (name, phone number, birth

date) in a line using the code below. (If no contact information is stored, do nothing,

go back to the menu options

1. **Search an existing contact:**

|  |
| --- |
| void findByBirth(struct Contact contacts[100], int count){  int month, i, monthInt;  char monthStr[2];  printf("Birth:");  scanf("%d", &month);  for(i=0; i<count; i++){  monthStr[0] = contacts[i].birthdate[4];  monthStr[1] = contacts[i].birthdate[5];  monthInt = atoi(monthStr);  if(monthInt == month)  printf("%s %s %s\n", contacts[i].name, contacts[i].phoneNumber, contacts[i].birthdate);  }  } |

Parameters:

* 1. Array of contacts
  2. Count of total contacts in array

Return:

void

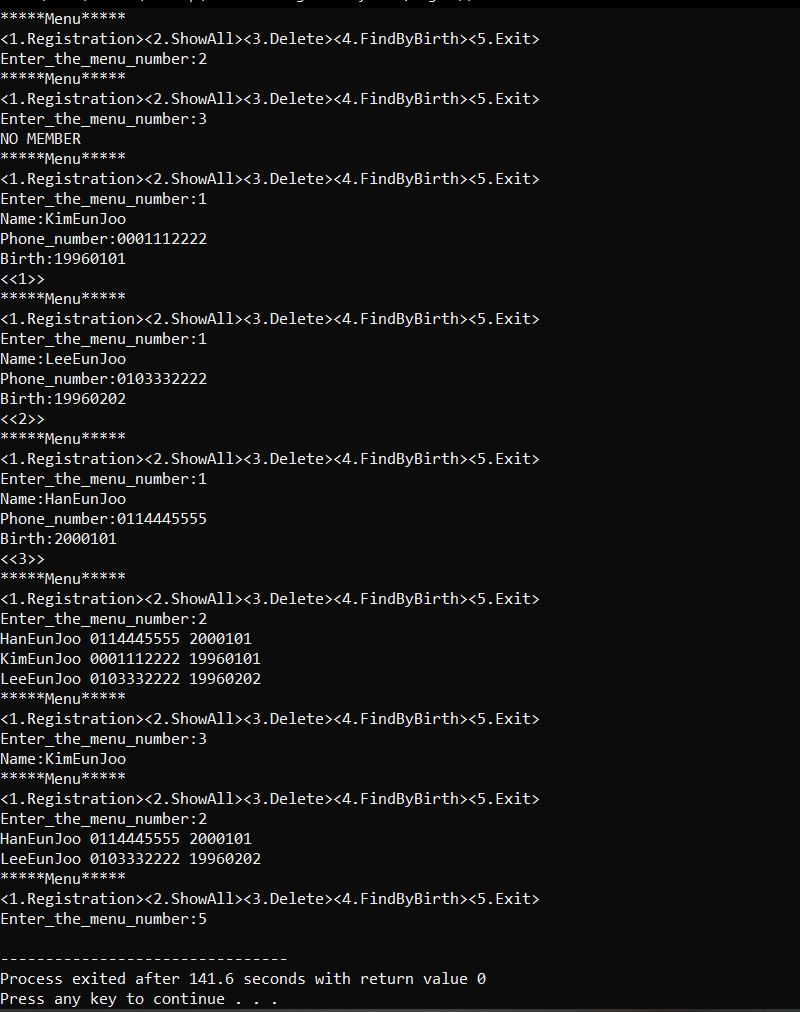
This function will take month of birth from user and display all contacts which are having same month as user input. Display the people’s information who were born in the month using the code

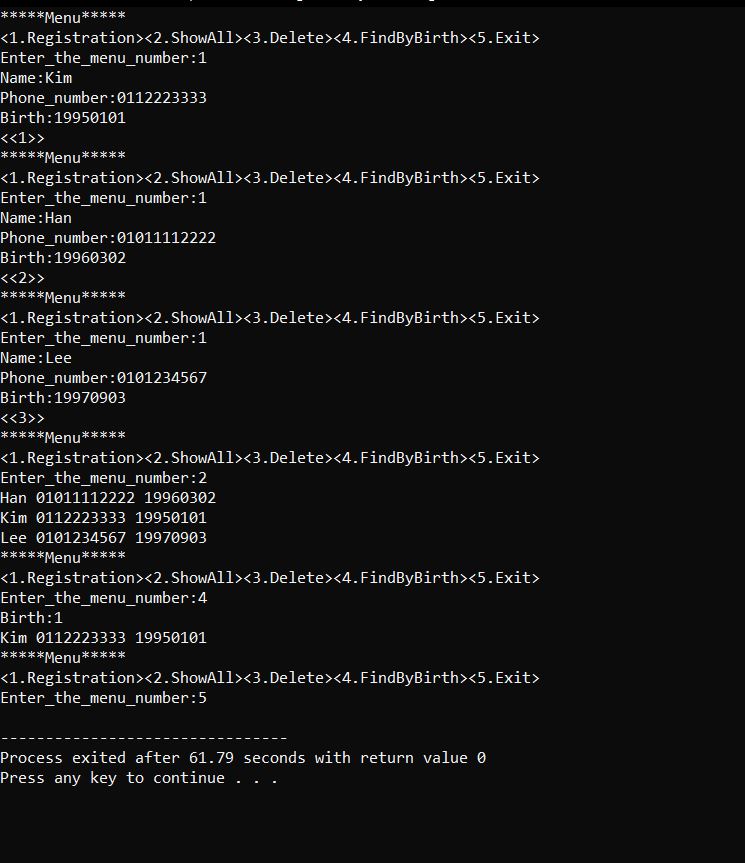
below. If no contact information is stored, do nothing, go back to the menu options. If

found multiple people, display their contact information by the order that they are

stored (i.e., order in ASCII code)

OUTPUTS





Stage – 2

Objective of stage – 2 is to implement system’s major functionality using pointers array,

**Structure of contact:**

|  |
| --- |
| struct Contact {  char \*name;  char \*phoneNumber;  char \*birthdate;  }; |

1. **Create a new contact:**

|  |
| --- |
| int registration(struct Contact \*\*contacts, int count){  contacts[count] = malloc(sizeof(struct Contact));  contacts[count]->name = malloc(sizeof(char) \*100);  contacts[count]->phoneNumber = malloc(sizeof(char) \*100);  contacts[count]->birthdate = malloc(sizeof(char) \*100);  printf("Name:");  scanf("%s", contacts[count]->name);  printf("Phone\_number:");  scanf("%s", contacts[count]->phoneNumber);  printf("Birth:");  scanf("%s", contacts[count]->birthdate);  int i, j, tempIndex;  struct Contact \*temp;  for(i=0; i<count; i++) {  tempIndex = i;  for(j=i+1; j<=count; j++){  if(strcmp(contacts[j]->name, contacts[tempIndex]->name) < 0)  tempIndex = j;  }  temp = contacts[tempIndex];  contacts[tempIndex] = contacts[i];  contacts[i] = temp;  }  printf("<<%d>>\n", ++count);  return count;  } |

Parameters:

1. Array of pointers of contacts
2. Count of total contacts in array

Return:

It will return new count after adding a new contact

This function will dynamically allocate memory to new struct then it will allocate memory to all the member of struct. Then it will take inputs from users which will contain username, phone number and birthdate and store this data into its cross-ponding index of array. Moreover, it will also sort array after each insertion.

1. **Delete a contact:**

|  |
| --- |
| int delete(struct Contact \*\*contacts, int count){  if(count == 0){  printf("NO MEMBER\n");  return count;  }  char name[21];  printf("Name:");  scanf("%s", name);  int i;  for(i=0; i<count; i++){  if(strcmp(contacts[i]->name, name) == 0){  int j;  free(contacts[i]->name);  free(contacts[i]->phoneNumber);  free(contacts[i]->birthdate);  free(contacts[i]);  for(j=i; j<count-1; j++)  contacts[j] = contacts[j+1];  --count;  }  }  return count;  } |

Parameters:

1. Array of pointers of contacts
2. Count of total contacts in array

Return:

It will return new count after adding a new contact

This function will take username from user and then it will free allocated memory to struct as well as all of its members. Afterwards, it will overwrite deleting struct will the struct next to it.

1. **Display all contacts:**

|  |
| --- |
| void showAll(struct Contact \*\*contacts, int count){  int i;  for(i=0; i<count; i++)  printf("%s %s %s\n", contacts[i]->name, contacts[i]->phoneNumber, contacts[i]->birthdate);  } |

Parameters:

1. Array of contacts
2. Count of total contacts in array

Return:

void

This function will display all contacts present in an array; order of printing will be same as they are inserted.

1. **Search an existing contact:**

|  |
| --- |
| void findByBirth(struct Contact \*\*contacts, int count){  int month, i, monthInt;  char monthStr[2];  printf("Birth:");  scanf("%d", &month);  for(i=0; i<count; i++){  monthStr[0] = contacts[i]->birthdate[4];  monthStr[1] = contacts[i]->birthdate[5];  monthInt = atoi(monthStr);  if(monthInt == month)  printf("%s %s %s\n", contacts[i]->name, contacts[i]->phoneNumber, contacts[i]->birthdate);  }  } |

Parameters:

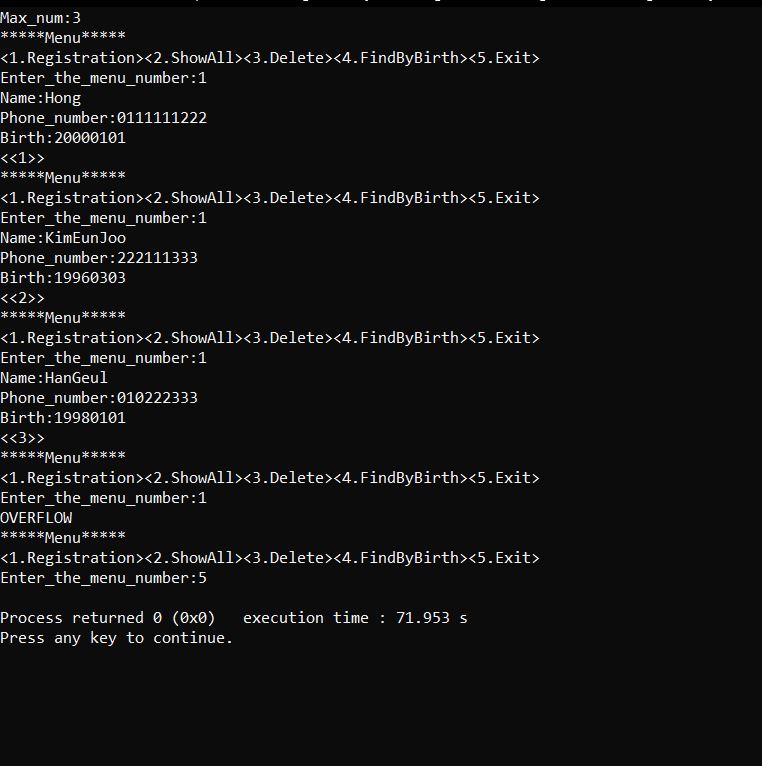
* 1. Array of contacts
  2. Count of total contacts in array

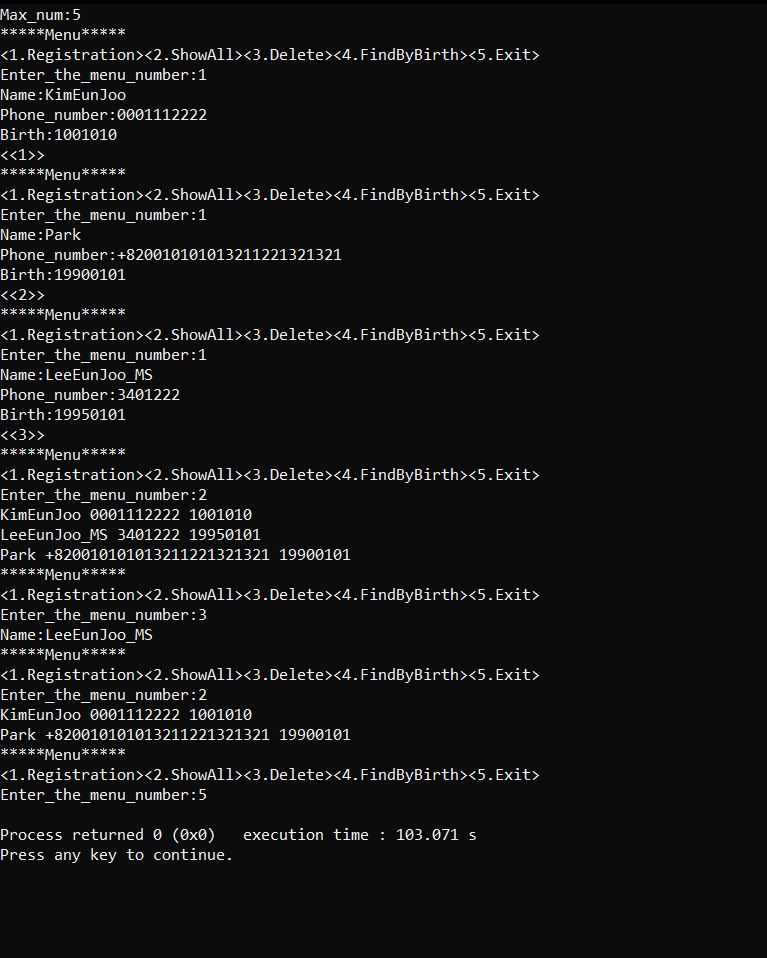
Return:

void

This function will take month of birth from user and display all contacts which are having same month as user input. Display the people’s information who were born in the month using the code below. If no contact information is stored, do nothing, go back to the menu options. If found multiple people, display their contact information by the order that they are stored (i.e., order in ASCII code)

OUTPUTS





Stage – 3

Objective of stage – 3 is to store and read contacts form text file.

**Structure of contact:**

|  |
| --- |
| struct Contact {  char \*name;  char \*phoneNumber;  char \*birthdate;  }; |

Most of the code between stage -2 and stage -3 is same. Only difference is of two methods.

1. **Register from text file:**

|  |
| --- |
| int regFromFile(struct Contact \*\*contacts, int count, int max){  FILE \* file = fopen("PHONE\_BOOK.txt", "r");  if (file == 0)  return count;  char temp[100];  while (fscanf(file, "%s", temp) != EOF){  if(count == max){  printf("OVERFLOW\n");  break;  }  contacts[count] = malloc(sizeof(struct Contact));  contacts[count]->name = malloc(sizeof(char) \*100);  contacts[count]->phoneNumber = malloc(sizeof(char) \*100);  contacts[count]->birthdate = malloc(sizeof(char) \*100);  strcpy(contacts[count]->name, temp);  fscanf(file, "%s", temp);  strcpy(contacts[count]->phoneNumber, temp);  fscanf(file, "%s", temp);  strcpy(contacts[count]->birthdate, temp);  count++;  } |

Parameters:

* 1. Array of pointers of contacts
  2. Count of total contacts in array
  3. Max number of allowed contacts

Return:

New count after insertion

This function will read contacts from text file one by one and then it will first check if array is not overflow then it will insert it into array of pointers.

1. **Write from text file:**

|  |
| --- |
| void writeToFile(struct Contact \*\*contacts, int count){  FILE \* file = fopen("PHONE\_BOOK.txt", "w");  if (file == 0)  return;  int i;  for(i=0; i<count; i++){  fputs(contacts[i]->name, file);  fputc(' ', file);  fputs(contacts[i]->phoneNumber, file);  fputc(' ', file);  fputs(contacts[i]->birthdate, file);  fputc('\n', file);  }  fclose(file);  } |

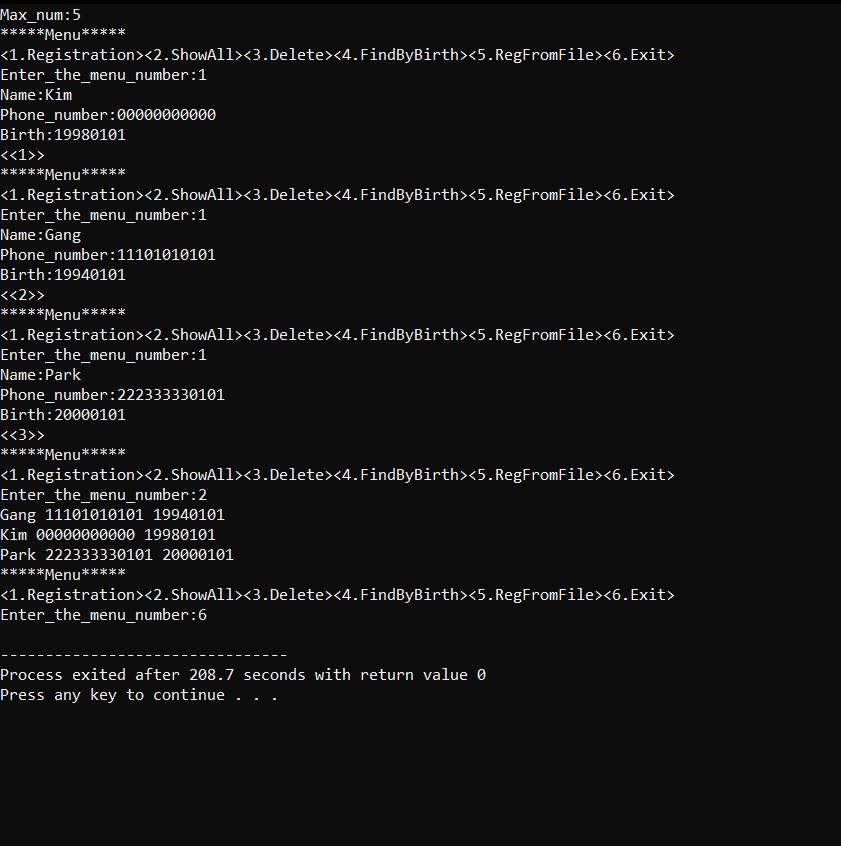
Parameters:

* 1. Array of pointers of contacts
  2. Count of total contacts in array

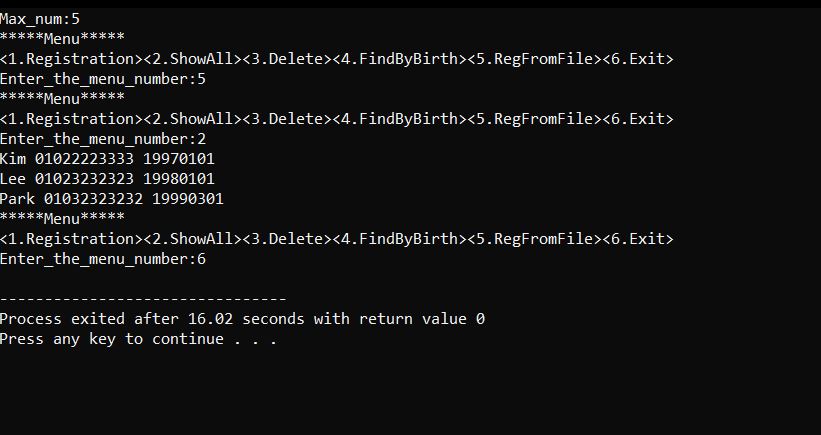
Return:

void

This function will write contacts back to the file for later use.

OUTPUTS

**File after first output:**



**File after second output:**

