Facebook application (Developer Guide)

General idea:

This is Facebook application which we need to read the community member's data from the file into memory and modify the data by inputs and display the necessary data on the screen.

I have used array of struct for this program. (dynamic)

Approaches:

1.Read the data from file to memory(database).

2.Insert new member's data to the file.

3.Find a member in the community by his family name/nickname

4.display people who are invited by the given nickname

5.display all the community members

- Libraries
- Structure
- Functions
- main

Libraries

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<windows.h> --> for system("cls");

#include<conio.h> --> mostly for getch(void) function and stream single char input
otherwise it will not be stream
```

Structure

The structure helps us to have structured data when we are reading from file so we can handle it.

Functions

- **int year ():**
   scans integer from user
   return int with the rang of 0<x<2022
  Usage:limiting the year input
  Call:In void create()

- **Int day ():**
   scans integer from user
  return int with the rang of 0<x<32
  Usage:limiting the day input
  Call:In void create()

- **Int month ():**
  scans integer from user and
   return int with the rang of 0<x<13
   Usage: limiting the month input
  Call:In void create()

- **char* check_if_only_chracter_alphabet(char* string):**
   It points for a string a checks char by char
  whether that string consist of non-alphabet if yes it scans string
  return: string pointer
  Usage:for checking the name, family name, birthplace and inviter's name and restricting them.

- **void create ();**
  The function will opens the file in append mode and asks the user to enter
  members data and calls **check_if_only_chracter_alphabet(char* string):**
  funtion for checking the data finally it will save the data from memory to file and closes the file
  and frees the allocated structure
  return:void
  Call: In main

- **void find_member(Member* member,int records);**
  The function points to the stack of memory which is allocated and passed by argument with the numbers of existence of records (we read each char in file once we hit \n record ++)
  we ask the user either a family name or nickname then we start comparing them with array of structure once we hit the
  strcm()==0
  match++, and we start printing it to screen;
  return:void
  Call:In main

- **void list_all(Member* member,int records);**
  same as find_member we get member pointer and number of records
  Then we display the "Name,Year,Birthplace" of the members
  return:void
  Call:In main

- **void delete_member(Member* member,int records)**
  This funtion will scans a string "nickname" from user
  we compare the user input with existence nicknames and try to find the index of that nickname in the structure
  then if we found the index we try to open a temp file and we store all the members data into a "temp.txt" file and once we hit that index that we was comparing before we skip it and do not store it again,
  finally delete the pervious file
  then we rename the temp as main ;
  return:void
  Call:In main

- **void search_invitor(Member* member,int records)**
  The function will receive a pointer pointing to the stack of array of struct and gets an ineteger as parameter.
  Scans inviter's name and compare it with the
  members->invitor
  if the input matches the invitros name
  we display is if not then
  did not match.
  Return:void
  Call:In main

Main

1. We open the file in **reading mode,** if open we continue, if not we make a file in **append mode**
2. We read char by char from the file once we hit the ‘\n’ we know we have a new record this will help us to allocate memory as much as we needed to.
3. Asking user:
   *Enter a new member's data
   *Find a member in the community
   *Delete a member from the community
   *List all the members who are invited by the same person
   *List all the Community Members
   *Exit the program
4. switch(menu)
5. We call the function for each option
   create()
   find_member(member,record_counter);
   delete_member(member,record_counter);
   search_invitor(member,record_counter);
   list_all(member,record_counter);
   Before passing the reocrds we check if it is not 0;
6. We keep looping these 6 steps untill user decided to leave