CSE-2003

J-COMPONENT REPORT

VIT UNIVERSITY, CHENNAI

Vandalur – Kelambakkam Road

NOVEMBER 2, 2020

“SMART CONTACT BOOK”

by

Shushant Singh (19BCE1620)
Apoorv Yadav (19BCE1163)
Aarohan B (19BCE1601)
Divyanu Baheti(19BCE1045)

A project report submitted to

Dr. PATTABIRAMAN V

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

CERTIFICATE

Certified that this project report entitled "***Smart Contact Book***" is a bonafide work of Divyanu, Apoorv, Shushant and Aarohan who carried out the "J"-Project work under my supervision and guidance for CSE-2003.

Dr. Pattabiraman V
SCHOOL OF COMPUTING SCIENCE AND ENGINEERING VIT University, Chennai
Chennai – 600 127.

ABSTRACT

A smart Contacts list which automatically sorts your contacts based on the time of the day. The contact who you frequently call during that time of the day will be positioned at the top.

We will make use of linked lists to link all the contacts together and will use the most efficient method of sorting by making use of all the concepts taught in this course (Data Structures and Algorithms).

For example:

During the day you always call your office colleague at 10:00 Am the system will automatically put that person on the top in the morning, and say at night you call your mother or father this system will automatically put that name on top.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project faculty, **Dr. Pattabiraman V,** for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

Finally, we would like to thank our deemed university, VIT Chennai, for providing us with the opportunity and facilities which ensured this project's completion.

1. INTRODUCTION

We will be using File handling and Linked List.

In computer science, a **linked list** is a linear collection of data elements whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence.

File Handling is the storing of data in a **file** using a program. In C programming language, the programs store results, and other data of the program to a **file** using **file handling** in C. Also, we can extract/fetch data from a **file** to work with it in the program.

2. DATASET USED

The project contains 2 datasets
The first one contains the main contacts. The file contains the following headings:

- First name
- Last name
- Phone number

➢ This dataset comes from a site Letter Hub.

➢ There are a total of 152 contacts in the dataset.

The Screenshot of the csv file:

	A	B	C	D	E	F	G
1	**Firstname**	**Lastname**	**Phone No.**				
2	Apoorv	Yadav	5048451427				
3	Ashish	Das	8103749840				
4	Shubham	Singh	8562644130				
5	Shushant	Singh	9079212010				
6	Divyanu	Baheti	5135494561				
7	Aarohan	B	4198006759				
8	Lenna	Paprocki	7739248565				
9	Donette	Foller	4088131105				
10	Simona	Morasca	6057944895				
11	Mitsue	Tollner	4108044694				
12	Leota	Dilliard	2154228694				
13	Sage	Wieser	6316773675				
14	Kris	Marrier	3102543084				
15	Minna	Amigon	4405797763				
16	Abel	Maclead	9568417216				
17	Kiley	Caldarera	6029536360				
18	Graciela	Ruta	9312357959				
19	Cammy	Albares	4143772880				
20	Mattie	Poquette	3133414470				
21	Meaghan	Garufi	8154265657				

Another csv is the Call Logs file that is generated after every calling. It contains the headings:

- First name
- Year
- Month
- Date
- Hour
- Minute
- Seconds

The screen shot of the file:

	A	B	C	D	E	F	G	H	I
1	First name	Year	Month	Date	Hour	Minute	Seconds		
2	Shushant	2020	11	1	16	33	27		
3	Shushant	2020	11	1	16	33	30		
4	Apoorv	2020	11	1	16	33	35		
5	Ashish	2020	11	1	16	33	41		
6	Ashish	2020	11	1	16	33	45		
7	Shushant	2020	6	2	18	53	55		
8	Ashish	2020	5	13	20	33	51		
9	Apoorv	2020	11	1	18	34	24		
10	Shushant	2020	11	1	18	35	41		
11	Erick	2020	10	29	10	56	45		
12	Aarohan	2020	3	13	23	19	57		
13	Divyanu	2020	2	19	2	45	45		
14	James	2020	11	24	12	51	48		
15	Kati	2020	10	16	16	28	12		
16	Erica	2020	9	23	17	34	23		
17	Dani	2020	1	30	8	23	28		

3. IMPLEMENTATION CODE

```
# include<stdio.h>
# include<conio.h>
# include<string.h>
# include<stdlib.h>
# include<time.h>
void menu();
  void adddetails();
  void searchdetails();
  void modifydetails();
  void deletedetails();
  void display();
  void call();
  void smartsortfunc();
  void load_data();
  void update_file();
  void sort();
  void old_record();
  struct contact {
    char lastname[20], firstname[20], mob_no[15];//Defining our Linked List//
    int count;
    struct contact * next;
  };
  typedef struct contact node;
  node * head, * newnode;
  void load_data()
  {
      FILE *fp;
      fp = fopen("contact_list.csv","r");
      if(fp == NULL)
      {
          printf("contact_list.csv file failed to open");
          exit(1);
      }
      char line[200];
      while(fgets(line,sizeof(line), fp))
      {
          node * ptr, * previous, * newnode;
          newnode = (node * ) malloc(sizeof(node));
          char *token;
          token = strtok(line, ",");
          for(int i=0;token!=NULL;i++)
          {
```

```
46.               if(i==0)
47.               {
48.                   strcpy(newnode -> firstname, token);
49.               }
50.               if(i==1)
51.               {
52.                   strcpy(newnode -> lastname, token);
53.               }
54.               if(i==2)
55.               {
56.                   int len = strlen(token)-1;
57.                   token[len] = '\0';
58.                   strcpy(newnode -> mob_no, token);
59.                   newnode->count=0;
60.               }
61.               token = strtok(NULL,",");
62.           }
63.             newnode -> next = NULL;
64.             if (head == NULL) {
65.               head = newnode;//If there is no linked list then add the details t
   o the head//
66.             } else {
67.               previous = ptr = head;
68.               //This loop sorts out the Contact in ascending order.//
69.               while (strcmp(newnode -> firstname, ptr -> firstname) > 0) {
70.                 previous = ptr;
71.                 ptr = ptr -
   > next; //The loops breaks if it is at the end of the linked list or //
72.                 if (ptr == NULL) break;//If the character of the new node is les
   s than the character of the new node ascii.//
73.               }
74.               if (ptr == previous) {
75.                 newnode -> next = head;//Insert at the 1st positon.//
76.                 head = newnode;
77.               } else if (ptr == NULL) {
78.                 previous -> next = newnode;//Insert in the last position//
79.               } else {
80.                 newnode -> next = ptr;
81.                 previous -> next = newnode;//Insert in the middle//
82.               }
83.             }
84.       }
85.       fclose(fp);
86.   }
87.   void old_record()
```

```
88.  {
89.      FILE *fp,*np;
90.      fp=fopen("call_log.csv","r");
91.      np=fopen("call_log1.csv","wb+");
92.      time_t t= time(NULL);
93.      struct tm tm = *localtime(&t);
94.      int cur_year = tm.tm_year+1900;
95.      int cur_month = tm.tm_mon + 1;
96.      if(fp==NULL)
97.      {
98.          printf("Failed to open file call_log.csv");
99.          exit(1);
100.     }
101.     char line[200];
102.     while(fgets(line,sizeof(line), fp))
103.     {
104.         char *token;
105.         char name[30];
106.         int year,month,date,min,sec,r_year,r_month,hour;
107.         token = strtok(line, ",");
108.         for(int i=0; token!=NULL; i++)
109.         {
110.             if(i==0)
111.             {
112.                 strcpy(name, token);
113.             }
114.             if(i==1)
115.             {
116.                 r_year=atoi(token);
117.             }
118.             if(i==2)
119.             {
120.                 r_month=atoi(token);
121.             }
122.             if(i==3)
123.             {
124.                 date = atoi(token);
125.             }
126.             if(i==4)
127.             {
128.                 hour=atoi(token);
129.             }
130.             if(i==5)
131.             {
132.                 min=atoi(token);
```

```
133.             }
134.             if(i==6)
135.             {
136.                 sec=atoi(token);
137.             }
138.             token = strtok(NULL,",");
139.         }
140.         year= r_year;
141.         month= r_month;
142.         if(cur_year == year && month< cur_month-6)
143.         {
144.             continue;
145.         }
146.         if(year < cur_year){
147.             month = month +6;
148.             if(month > 12)
149.             {
150.                 year =year +1;
151.                 month = month -12;
152.             }
153.             if(year<cur_year || (cur_year == year && month< cur_month)){
154.                 continue;
155.             }
156.         }
157.         fprintf(np,"%s,%d,%d,%d,%d,%d,%d\n",name,r_year,r_month,date,hour,min,
    sec);
158.     }
159.     fclose(fp);
160.     fclose(np);
161.     remove("call_log.csv");
162.     rename("call_log1.csv","call_log.csv");
163. }
164. void smartsortfunc()
165. {
166.     struct data{
167.         char name[30];
168.         int count;
169.     };
170.     FILE *fp;
171.     fp = fopen("call_log.csv","r");
172.     time_t t= time(NULL);
173.     struct tm tm = *localtime(&t);
174.     int cur_hour = tm.tm_hour;
175.     int cur_min = tm.tm_min;
176.     if(fp == NULL)
```

```
177.     {
178.         printf("call_log.csv file failed to open");
179.         exit(1);
180.     }
181.     node * ptr;
182.     for (ptr = head; ptr != NULL; ptr = ptr -> next){
183.         ptr->count=0;
184.     }
185.     char line[200];
186.     while(fgets(line,sizeof(line), fp))
187.     {
188.         char *token;
189.         char name[30];
190.         int hour,min;
191.         token = strtok(line, ",");
192.         for(int i=0;token!=NULL;i++)
193.         {
194.             if(i==0)
195.             {
196.                 strcpy(name, token);
197.             }
198.             if(i==4)
199.             {
200.                 hour=atoi(token);
201.             }
202.             if(i==5)
203.             {
204.                 min=atoi(token);
205.             }
206.             token = strtok(NULL,",");
207.         }
208.         if(cur_hour-1<=hour && hour<=cur_hour+1){
209.             if(hour==cur_hour-1 && min<cur_min)
210.             {
211.                 continue;
212.             }
213.             if(hour==cur_hour+1 && min>cur_min)
214.             {
215.                 continue;
216.             }
217.             node * ptr;
218.             ptr = head;
219.             while (strcmp(ptr -> firstname, name) != 0) {
220.                 ptr = ptr -> next;
221.                 if (ptr == NULL) break;
```

```
222.             }
223.             if (ptr != NULL) {
224.                 ptr -> count++;
225.             } else {
226.                 printf("No such record Found .......\n");
227.             }
228.         }
229.     }
230.     for (ptr = head; ptr != NULL; ptr = ptr -> next){
231.         printf("\nThe count of %s is %d",ptr->firstname,ptr->count);
232.     }
233.     fclose(fp);
234.     sort();
235.     printf("\n\t\t------------------------------\n");
236.     for (ptr = head; ptr != NULL; ptr = ptr -> next) {
237.         printf("\t\tFirst name: %s", ptr -> firstname);
238.         printf("\n\t\tLast name:%s", ptr -> lastname);
239.         printf("\n\t\tTelephone No.: %s", ptr -> mob_no);
240.         printf("\n\t\t------------------------------\n");
241.   }
242.     update_file();
243.     load_data();
244.     getch();
245.     menu();
246. }
247. int main() {
248.
249.     head = (node * ) malloc(sizeof(node));
250.     head = NULL;
251.     load_data();
252.     old_record();
253.     menu();
254.     return 0;
255.
256. }
```

CONCLUSION

- By the end of our project u will be getting a brand-new contact book software with much faster working and efficiency.
- It will be very user-friendly and will be very easy to access.
- It will save those little time of our users which will in hindsight save a lot time.

.