**DATA STRUCTURE ASSIGNMENT-1**

**Phone Book**

**#include<stdio.h>**

**#include<conio.h>**

**#include<string.h>**

**#include<stdlib.h>**

**struct person**

**{**

**char name[35];**

**long int mble\_no;**

**};**

**void menu();**

**void got();**

**void start();**

**void back();**

**void addrecord();**

**void listrecord();**

**void modifyrecord();**

**void deleterecord();**

**void searchrecord();**

**int main()**

**{**

**system("color 9f");**

**start();**

**return 0;**

**}**

**void back()**

**{**

**start();**

**}**

**void start()**

**{**

**menu();**

**}**

**void menu()**

**{**

**system("cls");**

**printf("\t\t\*\*\*\*\*\*\*\*\*\*WELCOME TO PHONEBOOK\*\*\*\*\*\*\*\*\*\*\*\*\*");**

**printf("\n\n\t\t\t MENU\t\t\n\n");**

**printf("\t1.Add New \t2.List \t3.Exit \n\t4.Modify \t5.Search\t6.Delete");**

**switch(getch())**

**{**

**case '1':**

**addrecord();**

**break;**

**case '2': listrecord();**

**break;**

**case '3': exit(0);**

**break;**

**case '4': modifyrecord();**

**break;**

**case '5': searchrecord();**

**break;**

**case '6': deleterecord();**

**break;**

**default:**

**system("cls");**

**printf("\nEnter 1 to 6 only");**

**printf("\n Enter any key");**

**getch();**

**menu();**

**}**

**}**

**void addrecord()**

**{**

**system("cls");**

**FILE \*f;**

**struct person p;**

**f=fopen("project","ab+");**

**printf("\n Enter name: ");**

**got(p.name);**

**printf("\nEnter phone no.:");**

**scanf("%ld",&p.mble\_no);**

**fwrite(&p,sizeof(p),1,f);**

**fflush(stdin);**

**printf("\nrecord saved");**

**fclose(f);**

**printf("\n\nEnter any key");**

**getch();**

**system("cls");**

**menu();**

**}**

**void listrecord()**

**{**

**struct person p;**

**FILE \*f;**

**f=fopen("project","rb");**

**if(f==NULL)**

**{**

**printf("\nfile opening error in listing :");**

**exit(1);**

**}**

**while(fread(&p,sizeof(p),1,f)==1)**

**{**

**printf("\n\n\n YOUR RECORD IS\n\n ");**

**printf("\nName=%s\nMobile no=%ld",p.name,p.mble\_no);**

**getch();**

**system("cls");**

**}**

**fclose(f);**

**printf("\n Enter any key");**

**getch();**

**system("cls");**

**menu();**

**}**

**void searchrecord()**

**{**

**struct person p;**

**FILE \*f;**

**char name[100];**

**f=fopen("project","rb");**

**if(f==NULL)**

**{**

**printf("\n error in opening\a\a\a\a");**

**exit(1);**

**}**

**printf("\nEnter name of person to search\n");**

**got(name);**

**while(fread(&p,sizeof(p),1,f)==1)**

**{**

**if(strcmp(p.name,name)==0)**

**{**

**printf("\n\tDetail Information About %s",name);**

**printf("\nName:%sMobile no:%ld",p.name,p.mble\_no);**

**}**

**else**

**printf("file not found");**

**}**

**fclose(f);**

**printf("\n Enter any key");**

**getch();**

**system("cls");**

**menu();**

**}**

**void deleterecord()**

**{**

**struct person p;**

**FILE \*f,\*ft;**

**int flag;**

**char name[100];**

**f=fopen("project","rb");**

**if(f==NULL)**

**{**

**printf("CONTACT'S DATA NOT ADDED YET.");**

**}**

**else**

**{**

**ft=fopen("temp","wb+");**

**if(ft==NULL)**

**printf("file opaning error");**

**else**

**{**

**printf("Enter CONTACT'S NAME:");**

**got(name);**

**fflush(stdin);**

**while(fread(&p,sizeof(p),1,f)==1)**

**{**

**if(strcmp(p.name,name)!=0)**

**fwrite(&p,sizeof(p),1,ft);**

**if(strcmp(p.name,name)==0)**

**flag=1;**

**}**

**fclose(f);**

**fclose(ft);**

**if(flag!=1)**

**{**

**printf("NO CONACT'S RECORD TO DELETE.");**

**remove("temp.txt");**

**}**

**else**

**{**

**remove("project");**

**rename("temp.txt","project");**

**printf("RECORD DELETED SUCCESSFULLY.");**

**}**

**}**

**}**

**printf("\n Enter any key");**

**getch();**

**system("cls");**

**menu();**

**}**

**void modifyrecord()**

**{**

**int c;**

**FILE \*f;**

**int flag=0;**

**struct person p,s;**

**char name[50];**

**f=fopen("project","rb+");**

**if(f==NULL)**

**{**

**printf("CONTACT'S DATA NOT ADDED YET.");**

**exit(1);**

**}**

**else**

**{**

**system("cls");**

**printf("\nEnter CONTACT'S NAME TO MODIFY:\n");**

**got(name);**

**while(fread(&p,sizeof(p),1,f)==1)**

**{**

**if(strcmp(name,p.name)==0)**

**{**

**printf("\n Enter name:");**

**got(s.name);**

**printf("\nEnter phone no:");**

**scanf("%ld",&s.mble\_no);**

**fseek(f,-sizeof(p),SEEK\_CUR);**

**fwrite(&s,sizeof(p),1,f);**

**flag=1;**

**break;**

**}**

**fflush(stdin);**

**}**

**if(flag==1)**

**{**

**printf("\n your data id modified");**

**}**

**else**

**{**

**printf(" \n data is not found");**

**}**

**fclose(f);**

**}**

**printf("\n Enter any key");**

**getch();**

**system("cls");**

**menu();**

**}**

**void got(char \*name)**

**{**

**int i=0,j;**

**char c,ch;**

**do**

**{**

**c=getch();**

**if(c!=8&&c!=13)**

**{**

**\*(name+i)=c;**

**putch(c);**

**i++;**

**}**

**if(c==8)**

**{**

**if(i>0)**

**{**

**i--;**

**}**

**// printf("h");**

**system("cls");**

**for(j=0;j<i;j++)**

**{**

**ch=\*(name+j);**

**putch(ch);**

**}**

**}**

**}while(c!=13);**

**\*(name+i)='\0';**

**}**

**Q2**

**Multiple Queues:-**

Multiple queues are queue which maintain more than 1 queue at a time in a single array. We might use one dimensional array or multidimensional array to illustrated a multiple queue. Each of queues contains n elements that are mapped to a liner array of m elements. in order to maintain two queues there should be two Front and two rear of the two queues Both the queues can grow up to any extent from 1st position to maximum. Hence there should be one or more variable to keep track of the total number of values stored. The overflow condition will appear. If count becomes equal to or greater than array size and the underflow, condition from empty queue will appear if count =0

Structure multiqueue

{

Int Front 0. Front 1;

Int Rear 0. Rear 1;

Int num;

Int count;

};

**Multiple stack:-**

Multiple Stack computers have two or more stacks supported by the instruction set. One stack is usually intended to store return addresses, the other stack is for expression evaluation and subroutine parameter passing. Multiple stacks allow separating control flow information from data operands.

#include <stdio.h>

#define size 20

int array[size];

int top1 = -1;

int top2 = size;

void push1 (int d)

{

if (top1 < top2 - 1)

{

top1++;

array[top1] = d;

}

else

{

printf ("Stack is full");

}

}

void push2 (int d)

{

if (top1 < top2 - 1)

{

top2--;

array[top2] = d;

}

else

{

printf ("Stack is full..\n");

}

}

void pop1 ()

{

if (top1 >= 0)

{

int p = array[top1];

top1--;

printf ("%d is being popped from Stack 1\n", p);

}

else

{

printf ("Stack is Empty \n");

}

}

void pop2 ()

{

if (top2 < size)

{

int p = array[top2];

top2--;

printf ("%d is being popped from Stack 1\n",p);

}

else

{

printf ("Stack is Empty!\n");

}

}

void display\_stack1 ()

{

int i;

for (i = top1; i >= 0; --i)

{

printf ("%d ", array[i]);

}

printf ("\n");

}

void display\_stack2 ()

{

int i;

for (i = top2; i < size; ++i)

{

printf ("%d ", array[i]);

}

printf ("\n");

}

int main()

{

int ar[size];

int i;

int n;

printf ("We can push a total of 20 values\n");

for (i = 1; i <= 10; ++i)

{

push1(i);

printf ("Value Pushed in Stack 1 is %d\n", i);

}

for (i = 11; i <= 20; ++i)

{

push2(i);

printf ("Value Pushed in Stack 2 is %d\n", i);

}

display\_stack1 ();

display\_stack2 ();

printf ("Pushing Value in Stack 1 is %d\n", 11);

push1 (11);

n = top1 + 1;

while (n)

{

pop1 ();

--n;

}

pop1 ();

return 0;

}