**ProJect Using Dsa**

**-INDEX**

1.Problem Statement

2.Functions / Modules

3.List of Errors Encountered while coding the project

4.Key or challenging logic in the project

5.Prerequisites

6.Project Code

7. Elaborate Sample Input and Output Screenshots

**-PROBLEM STATEMENT**

To make a basic music player using the knowledge of various Data structures .

Data structures used include a doubly linked list, stacks and queues.

File handling to read and write songs has also been used.

**-FUNCTIONS**

1. add\_file () –

Function to work on playslist.txt.

2. add\_song () –

Function that adds songs to the list using a linked list.

3. add\_node\_file () –

Function that adds songs to the playlist to linked list from the data passed in addplaylist() function.

4. delete\_song () –

Function to delete song from text file playlist.txt.

5. delete\_node () –

Function that deletes the last song from the input linked list.

6. print\_playlist () –

Function that displays the input songs of the playlist.

7. count\_song () –

Function that tracks the number of inputs in the linked list.

8. del\_by\_od () –

Function that deletes songs from the linked list using the position of that song.

9. search1() –

Function that takes song input and linearly searches through the linked list and finds matching case.

10. push() –

Function that pushes the last played track of play() function into a stack to store and create a recently played list.

11. display() -

Function to display the stack generated in push() function.

12. play() -

Function to search input song and show if it can be played. It then passes the song to push() function to be added to recently played list.

13. recent() -

Function that calls display() function.

14. topelement() –

Function that displays the last played song.

15. addplaylist() –

Function that opens text file playlist.txt and passes data to add\_node\_file() function.

16. del\_search() –

Function to search input song and delete it from the list.

17. deletemenu() –

Function to invoke del\_search() or del\_pos() functions depending on user input.

18. main() –

Function that invokes all other functions of the project based on user defined input.

**-KEY OR CHALLENGING LOGIC**

The major part of the challenge included making a doubly linked list that would accept string data and store it efficiently. Involving the usage of file handling was another challenge. To retrieve songs from a pre-made list and add it to the linked list. It also required us to be able to write the new input songs to the file. All changes in the program required to be reflected on the text file.

**-PREQUISITES**

To use the File handling part of the program a text file of name

“MyplayList.txt” with certain data elements is required.

Sample data elements-

Kabir\_Playlist

Nusrat Fateh Ali khan

Kishor kumar

Arjit\_singh

**-PROJECT CODE**

#include<bits/stdc++.h>

#include<string.h>

#include<stdlib.h>

#include<stdio.h>

#include<fstream>

using namespace std;

  /\*   Frame work

  \*/

     // 1.. create the doublly linked list

struct node

{

char song[100];

 struct node \* prev;

 struct node \* next;

}\*top ,\*temp,\*top1;

// 2.. create the playlist in text format

void add\_file(char st[]){

fstream f1;

f1.open("MyplayList.txt",ios::out|ios::app);

f1<<st<<endl;

f1.close();

  }

//3.. add song using double linkedlist

void add\_song(struct node \* first){

  char sng[100];

  while(first->next!=NULL){

    first=first->next;

  }

  first->next=(struct node\*)malloc(sizeof(struct node));

first->prev = first;

first=first->next;

cout<<"\n\a\a\a\aEnter The Song name :-";

scanf("%s",&sng);

strcpy(first->song,sng);

add\_file(sng);

first->next=NULL;

}

//4.. add song to playlist to linklsit

void add\_node\_file(struct node \*first,string a){

 while(first->next!=NULL){

    first=first->next;

  }

 first->next=(struct node\*)malloc(sizeof(struct node));

first->prev = first;

first=first->next;

strcpy(first->song,a.c\_str());

first->next=NULL;

}

// 5.. delete the song from playlist

void delete\_song(char a[]){

fstream f1,f2;

string line;

bool found=0;

f1.open("MyplayList.txt",ios::in|ios::out);

f2.open("temp.txt",ios::in|ios::out);

while(!f1.eof())

{

getline(f1,line);

if(strcmp(a,line.c\_str())!=0)

  f2<<line<<endl;

  else if(strcmp(a,line.c\_str())==0)

         found=1;

}

f1.close();

f2.close();

remove("MyplayList.text");

rename("temp.txt","MyplayList.text");

if(!found)

{

  cout<<"#Song Not Found..."<<endl;

}

else {

cout<<"=>...Song has been Deleted..."<<endl;

}

}

//6.. delete last song from linklist

void delete\_node(struct node \*first)

{

while((first->next)->next!=NULL)

{

  first=first->next;

}

struct node \* temp;

temp=(first->next)->next;

first->next=NULL;

free(temp);

cout<<"..Deleted.."<<endl;

}

//7.. print the song from the playlist

void print\_playlist(struct node \*first)

{

cout<<"\nPlayList Name :-";

while(first->next!=NULL)

{

cout<<first->song<<endl;

first=first->next;

}

cout<<first->song<<endl;

}

//8..  count the total song

void count\_song(struct node \*first)

{

    int i=0;

    while (first->next!=NULL)

    {

        first=first->next;

        i++;

    }

    i++;

    cout<<"\nTotal songs :- "<<i-1<<endl;

}

// 9 .. for delete the song from the list

struct node \*del\_by\_od(struct node \* pointer ,int pos)

{

struct node \*n1,\*prev1,\*temp;

prev1=(struct node\*)malloc(sizeof(node));

temp=(struct node\*)malloc(sizeof(node));

int i=0;

// if we have to delete first song

if(pos==1)

{

temp=pointer;

delete\_song(temp->song);

pointer=pointer->next;

pointer->prev=NULL;

free(temp);

cout<<"\n=>The list has been updated\n\nUse the display function to check\n";

return pointer;

}

// if we have to delete any pos song

while(i<pos-1)

{

prev1=pointer;

pointer=pointer->next;

i++;

}

// if song is at last pos

if(pointer->next==NULL)

{

  temp=pointer;

  delete\_song(temp->song);

prev1->next->prev=NULL;

prev1->next=NULL;

free(temp);

cout<<"\n=>The list has been updated\n\nUse the display function to check\n";

}

temp=pointer;

delete\_song(temp->song);

prev1->next=temp->next;

temp->next->prev=prev1;

free(temp);

cout<<"\n=>The list has been updated\n\nUse the display function to check\n";

}

//10. searching a particular song

void search1(struct node\* first)

{

char song[100];

cout<<"\n\a\a\a\aEnter song To be Searched- ";

scanf("%s",&song);

int fl=0;

while(first!=NULL)

{

if(strcmp(first->song,song)==0)

{

         cout<<"\n\a\a\a\a#Song Found"<<endl;

         fl++;

            break;

}

else

{

first=first->next;

}

}

if(fl==0)

{

 cout<<"\n\a\a\a\aError:-> ..Song Not found.."<<endl;

}

}

//12.. for initaillize NULL to top

void create()

{

  top=NULL;

}

// 13..

void push(char data[])

{

if(top==NULL)

{

top=(struct node\*)malloc(sizeof(node));

top->next=NULL;

strcpy(top->song,data);

}

else if(strcmp(top->song,data)!=0)

{

temp=(struct node \*)malloc(sizeof(node));

temp->next=top;

strcpy(temp->song,data);

top=temp;

}

}

//14. display the recent song played

void display()

{

    top1 = top;

    if (top1 == NULL)

    {

        printf("\n\a\a\a\a..Oh..NO recently played tracks.\n");

        return;

    }

    printf("\n\a\a\a\a#..Recently played tracks: -\n");

    while (top1 != NULL)

    {

        printf("%s", top1->song);

        printf("\n");

        top1 = top1->next;

    }

 }

// 15..for the play the song

void play(struct node \*first)

{

char song[100];

print\_playlist(first);

cout<<"\n\a\a\a\aHey!.Choose Song you wish to play:-";

scanf("%s",song);

int fl=0;

while(first!=NULL)

{

if(strcmp(first->song,song)==0)

{

cout<<"\n\a\a\a\a.Now playing....Enjoy ..:=>  "<<song<<endl;

fl++;

push(song);

break;

}

else

{

first=first->next;

}

if(fl==0)

{

cout<<"\n\a\a\a\aError:Song Not Found"<<endl;

}

}

}

//16.. to show the recent song play

void recent()

{

display();

}

//17.. To display the last played song

void topelement()

{

top1=top;

if(top1==NULL)

{

cout<<"\n\a\a\a\a# NO last Played Tracks .\n";

return;

}

cout<<"\n=>Last Played Song-"<<top->song<<endl;

}

void add\_playlist(struct node \*start)

{

fstream f1;

string line;

f1.open("MyPlayList.txt",ios::in);

while(!f1.eof())

{

getline(f1,line);

add\_node\_file(start,line);

}

cout<<"=>PlayList Added"<<endl;

f1.close();

}

//18.. To delete the song from the playlist

void del\_search(struct node \* start)

{

char song[100];

print\_playlist(start);

cout<<"\n\a\a\a\aChoose Song You wish to delete:- ";

scanf("%s",&song);

int fl=0;

while(start!=NULL)

{

  if(strcmp(start->song,song)==0)

  {

    cout<<"\n\a\a\a#Song Found"<<endl;

    struct node \*temp;

    temp=(struct node \*)malloc(sizeof(node));

    temp=start;

    delete\_song(temp->song);

    temp->prev->next=temp->next;

    temp->next->prev=temp->prev;

    free(temp);

    fl++;

    break;

  }

  else {

    start=start->next;

  }

}

if(fl==0)

{

  cout<<"\n\a\a\a#Song Not Found"<<endl;

}

}

//19.. invoke delete by search and delete by position

void deleteMenu(struct node \*start)

{

int c;

cout<<"Delete Song?\n1.By Search\n2.By position"<<endl;

cout<<"\a\a\a\aEnter Your Choice:-";

cin>>c;

switch(c)

{

    case 1:del\_search(start);

    break;

    case 2:int pos;

            printf("\n\a\a\a\aEnter the song position : ");

            scanf("%d",&pos);

            del\_by\_od(start,pos-1);

    break;

}

}

//20.. main function (entry point)

int main()

{

int choice;

char song[100];

struct node \* start,\*hold;

start=(struct node \*)malloc(sizeof(struct node));

cout<<"\t\t\t\a\a\a\a\*\*WELCOME\*\*"<<endl;

cout<<"\n\*\*please use '\_' for space."<<endl;

 cout<<"\n\n\a\a\a\aEnter your playlist name-  ";

cin.getline(start->song,100);

start->next=NULL;

hold=start;

create();

    do{

      cout<<"\n\a\a\a\a...........................................................................................................\n";

cout<<"\n\a\a\a\a..................................................................................................................\n";

        cout<<"\n1.Add  New Song\n2.Delete Song\n3.Display Entered Playlist\n4.Total Songs\n5.Search Song\n6.Play Song\n7.Recently Played List\n8.Last Played\n9. Add From File\n10.Exit"<<endl;

        cout<<("\n\a\a\a\aEnter your choice- ");

        cin>>choice;

        switch(choice)

        {

            case 1:add\_song(start);

            break;

            case 2:deleteMenu(start);

            break;

            case 3:print\_playlist(start);

            break;

            case 4:count\_song(hold);

            break;

            case 5:search1(start);

            break;

            case 6:play(start);

            break;

            case 7:recent();

            break;

            case 8:topelement();

            break;

            case 9:add\_playlist(start);

            break;

            case 10:cout<<"\n\a\a\a\aThank you..";exit(0);

        }

    }while(choice!=11);

}

**-Sample Input and Output Screenshots**

