DATA STRUCTURES

PROJECT-REPORT

By: Nidadavolu Kundan Sai raj

ABSTRACT

This Project presents a Menu driven program containing all the Data Structures in a single program. The Programming language used is **C-language** and the compiler **Dev C++**. The Data Structures included in the Program are:

1. SINGLY-LINKED-LISTS
2. DOUBLY-LINKED-LISTS
3. CIRCULR-LINKED-LISTS
4. STACKS
5. QUEUES

ACKNOWLEDGEMENT

This project was worked under the guidance of Doctor Naresh Kar sir.

We want to Thank sir for sharing his valuable Time, Experience and Knowledge which made this project possible.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.NO | TOPIC | Page No. |
| 1) | Source Code | 5 - 29 |
| 2) | Output Screen-shots | 30 - 33 |
| 3) | DS Applicative Programming | 34 - 41 |
| 4) | Bibliography | 42 |

SOURCE CODE

#include<stdio.h>

#include<stdlib.h>

struct ll

{

int data;

struct ll \*next;

}\*f1,\*rear;

struct dll

{

int data;

struct dll \*next;

struct dll \*prev;

}\*f2;

// SINGLY-LINKED-LIST-FUNCTIONS

void createll(int A[],int n)

{

int i;

struct ll \*t,\*last;

f1=(struct ll \*)malloc(sizeof(struct ll));

f1->data=A[0];

f1->next=NULL;

last=f1;

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

{

t=(struct ll \*)malloc(sizeof(struct ll));

t->data=A[i];

t->next=NULL;

last->next=t;

last=t;

}

}

void displayll(struct ll \*p)

{

while(p)

{

printf("%d\t",p->data);

p=p->next;

}

}

void newnodell(struct ll \*p,int y)

{

int x,i;

struct ll \*t,\*u;

t=(struct ll \*)malloc(sizeof(struct ll));

t->data=y;

printf("\n");

printf("Enter 1 to add a node at the begining\n");

printf("Enter 2 to add a node at the end\n");

printf("Enter 3 to add a node at a specified Location\n");

printf("Enter a number: ");

scanf("%d",&x);

switch(x)

{

case 1:

printf("Adding a new node at the begining!\n");

t->next=f1;

f1=t;

break;

case 2:

printf("Adding a node at the end!\n");

while(p->next!=NULL)

{

p=p->next;

}

t->next=NULL;

p->next=t;

break;

case 3:

printf("Adding a node at a specified location!\n");

int j;

printf("Enter the location: ");

scanf("%d",&j);

for(i=0;i<j-1;i++)

{

u=p;

p=p->next;

}

u->next=t;

t->next=p;

break;

default:

printf("Wrong Number Entered!\n");

break;

}

}

void delnodell(struct ll \*p)

{

int x,y,z,i;

struct ll \*l;

printf("\n");

printf("Enter 1 to Delete the first node\n");

printf("Enter 2 to Delete the Last node\n");

printf("Enter 3 to Delete a specified node\n");

printf("Enter a number: ");

scanf("%d",&x);

switch(x)

{

case 1:

printf("\n");

printf("Deleting the first node\n");

f1=f1->next;

y=p->data;

p->next=NULL;

free(p);

printf("%d data has been deleted\n",y);

break;

case 2:

printf("\n");

printf("Deleting the last node\n");

while(p->next!=NULL)

{

l=p;

p=p->next;

}

l->next=NULL;

y=p->data;

free(p);

printf("%d has been deleted\n",y);

break;

case 3:

printf("\n");

printf("Deleting a node at a specified location\n");

printf("Enter the node location to be deleted: ");

scanf("%d",&z);

for(i=0;i<z-1;i++)

{

l=p;

p=p->next;

}

l->next=p->next;

p->next=NULL;

y=p->data;

free(p);

printf("%d data has been deleted\n",y);

break;

default:

printf("Wrong Number Entered\n");

break;

}

}

void revll(struct ll \*p)

{

struct ll \*r,\*s;

r=s=NULL;

while(p)

{

r=s;

s=p;

p=p->next;

s->next=r;

}

f1=s;

}

// SINGLY-LINKED-LIST-FUNCTIONS-END

// DOUBLY-LINKED-LIST-FUNCTIONS

createdll(int A[],int n)

{

struct dll \*t,\*last;

int i;

f2=(struct dll \*)malloc(sizeof(struct dll));

f2->data=A[0];

f2->prev=NULL;

f2->next=NULL;

last=f2;

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

{

t=(struct dll \*)malloc(sizeof(struct dll));

t->data=A[i];

t->next=NULL;

t->prev=last;

last->next=t;

last=t;

}

}

void displaydll(struct dll \*p)

{

while(p)

{

printf("%d\t",p->data);

p=p->next;

}

}

void newnodedll(struct dll \*p,int x)

{

struct dll \*t,\*l;

int i,z,v;

t=(struct dll \*)malloc(sizeof(struct dll));

t->data=x;

printf("Enter 1 to add a node at the begining\n");

printf("Enter 2 to add a node at the end\n");

printf("Enter 3 to add a node at a specified location\n");

printf("Enter a number: ");

scanf("%d",&z);

switch(z)

{

case 1:

printf("\n");

printf("Adding a Node at the Begining!\n");

t->next=f2;

t->prev=NULL;

f2=t;

break;

case 2:

printf("\n");

printf("Adding a Node at the End!\n");

while(p->next!=NULL)

{

p=p->next;

}

t->prev=p;

t->next=NULL;

p->next=t;

break;

case 3:

printf("\n");

printf("Adding a Node at a Specified location!\n");

printf("Enter the position: ");

scanf("%d",&v);

for(i=0;i<v;i++)

{

l=p;

p=p->next;

}

t->next=p;

p->prev=t;

t->prev=l;

l->next=t;

break;

default:

printf("\n");

printf("Wrong-Number-Entered!\n");

break;

}

}

void delnodedll(struct dll \*p)

{

int x,z,t,i;

printf("Enter 1 to Delete the first node\n");

printf("Enter 2 to Delete the Lat node\n");

printf("Enter 3 to Delete a specified node\n");

printf("\nEnter the Number: ");

scanf("%d",&x);

switch(x)

{

case 1:

printf("Deleting the First-Node!\n");

f2=f2->next;

t=p->data;

p->next->prev=NULL;

p->next=NULL;

free(p);

break;

case 2:

printf("Deleting the Last-Node!\n");

while(p->next!=NULL)

{

p=p->next;

}

t=p->data;

p->prev->next=NULL;

p->prev=NULL;

free(p);

break;

case 3:

printf("Deleting a Specified Node location!");

printf("Enter the position: ");

scanf("%d",&z);

for(i=0;i<z;i++)

{

p=p->next;

}

p->next->prev=p->prev;

p->prev->next=p->next;

t=p->data;

free(p);

break;

default:

printf("Wrong-Number Entered!");

break;

}

printf("%d data has been deleted!\n",t);

}

// END OF DOUBLY-LINKED-LISTS FUNCTIONS

// CIRCULAR-LINKED-LIST-PROGRAMMES

void createcll(int A[],int n)

{

int i;

struct ll \*t,\*last;

f1=(struct ll \*)malloc(sizeof(struct ll));

f1->data=A[0];

f1->next=NULL;

last=f1;

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

{

t=(struct ll \*)malloc(sizeof(struct ll));

t->data=A[i];

t->next=f1;

last->next=t;

last=t;

}

}

void displaycll(struct ll \*p)

{

do

{

printf("%d\t",p->data);

p=p->next;

}while(p!=f1);

}

void newnodecll(struct ll \*p,int y)

{

int x,i;

struct ll \*t,\*u;

t=(struct ll \*)malloc(sizeof(struct ll));

t->data=y;

printf("\n");

printf("Enter 1 to add a node at the begining\n");

printf("Enter 2 to add a node at the end\n");

printf("Enter 3 to add a node at a specified Location\n");

printf("Enter a number: ");

scanf("%d",&x);

switch(x)

{

case 1:

printf("Adding a new node at the begining!\n");

while(p->next!=f1)

{

p=p->next;

}

p->next=t;

t->next=f1;

f1=t;

break;

case 2:

printf("Adding a node at the end!\n");

while(p->next!=NULL)

{

p=p->next;

}

t->next=f1;

p->next=t;

break;

case 3:

printf("Adding a node at a specified location!\n");

int j;

printf("Enter the location: ");

scanf("%d",&j);

for(i=0;i<j-1;i++)

{

u=p;

p=p->next;

}

u->next=t;

t->next=p;

break;

default:

printf("Wrong Number Entered!\n");

break;

}

}

void delnodecll(struct ll \*p)

{

int x,y,z,i;

struct ll \*l,\*t;

printf("\n");

printf("Enter 1 to Delete the first node\n");

printf("Enter 2 to Delete the Last node\n");

printf("Enter 3 to Delete a specified node\n");

printf("Enter a number: ");

scanf("%d",&x);

switch(x)

{

case 1:

printf("\n");

printf("Deleting the first node\n");

t=f1;

while(t->next!=f1)

{

t=t->next;

}

f1=f1->next;

t->next=f1;

y=p->data;

p->next=NULL;

free(p);

printf("%d data has been deleted\n",y);

break;

case 2:

printf("\n");

printf("Deleting the last node\n");

while(p->next!=NULL)

{

l=p;

p=p->next;

}

l->next=f1;

p->next=NULL;

y=p->data;

free(p);

printf("%d has been deleted\n",y);

break;

case 3:

printf("\n");

printf("Deleting a node at a specified location\n");

printf("Enter the node location to be deleted: ");

scanf("%d",&z);

for(i=0;i<z-1;i++)

{

l=p;

p=p->next;

}

l->next=p->next;

p->next=NULL;

y=p->data;

free(p);

printf("%d data has been deleted\n",y);

break;

default:

printf("Wrong Number Entered\n");

break;

}

}

// END OF CIRCULAR-LINKED-LIST-PROGRAMES

// STACK FUNCTIONS

void createstack(int A[],int n)

{

struct ll \*t,\*l;

int i;

f1=(struct ll \*)malloc(sizeof(struct ll));

f1->data=A[0];

f1->next=NULL;

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

{

t=(struct ll \*)malloc(sizeof(struct ll));

t->next=f1;

t->data=A[i];

f1=t;

}

}

void displaysll(struct ll \*p)

{

while(p)

{

printf("%d\n",p->data);

p=p->next;

}

}

void push(int x)

{

struct ll \*t;

t=(struct ll \*)malloc(sizeof(struct ll));

t->data=x;

t->next=f1;

f1=t;

}

void pop(struct ll \*p)

{

f1=f1->next;

int x;

x=p->data;

printf("%d has been poped!\n",p->data);

}

// END OF STACK-PROGRAMES

// QUEUES-PROGRAMES

void createqll(int A[],int n)

{

struct ll \*t;

int i;

f1=(struct ll \*)malloc(sizeof(struct ll));

f1->data=A[0];

f1->next=NULL;

rear=f1;

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

{

t=(struct ll \*)malloc(sizeof(struct ll));

t->data=A[i];

t->next=NULL;

rear->next=t;

rear=t;

}

}

void displayqll(struct ll \*p)

{

while(p)

{

printf("%d\t",p->data);

p=p->next;

}

}

void newque(int x)

{

struct ll \*t;

t=(struct ll \*)malloc(sizeof(struct ll));

t->data=x;

t->next=NULL;

rear->next=t;

rear=t;

}

void delque(struct ll \*p)

{

int x;

f1=f1->next;

x=p->data;

p->next=NULL;

free(p);

}

int main()

{

int x,y,t;

int A[5]={5,3,8,1,2};

int n=5;

printf("Enter 1 for Singly-Linked-List\n");

printf("Enter 2 for Doubly-Linked-list\n");

printf("Enter 3 for Circular-Linked-list\n");

printf("Enter 4 for STACK\n");

printf("Enter 5 for QUEUES\n");

printf("Enter a number: ");

scanf("%d",&x);

switch(x)

{

case 1:

printf("\n");

printf("SINGLY-LINKED-LIST PROGRAMES\n");

createll(A,n);

printf("The Exsisting Linked-List\n");

displayll(f1);

printf("\n");

printf("\nEnter 1 to Add a new Node\n");

printf("Enter 2 to Delete a ndoe\n");

printf("Enter 3 to Reverse a Linked-List\n");

printf("Enter 4 for no changes and EXIT\n");

printf("Enter a number: ");

scanf("%d",&y);

switch(y)

{

case 1:

printf("\n");

printf("Adding a new node\n");

printf("Enter the data to be added: ");

scanf("%d",&t);

newnodell(f1,t);

displayll(f1);

break;

case 2:

printf("\n");

printf("Deleting a node\n");

delnodell(f1);

displayll(f1);

break;

case 3:

printf("\n");

printf("Reversing a Linked List\n");

revll(f1);

displayll(f1);

break;

case 4:

printf("\n");

printf("NO-CHANGES!\n");

displayll(f1);

break;

default:

printf("\n");

printf("Wrong-Number Entered\n");

break;

}

break;

case 2:

printf("\n");

printf("DOUBLY-LINKED-LIST-PROGRAMES\n");

createdll(A,n);

printf("The Existing Doubly-Linked-list\n");

displaydll(f2);

printf("\n");

printf("Enter 1 to Add a node\n");

printf("Enter 2 to delete a node\n");

printf("Enter 3 for no changes and EXIT\n");

printf("Enter a number: ");

scanf("%d",&y);

switch(y)

{

case 1:

printf("\n");

printf("Adding a New Node!\n");

printf("Enter the data to be Added: ");

scanf("%d",&t);

newnodedll(f2,t);

displaydll(f2);

break;

case 2:

printf("\n");

printf("Deleting a Node!\n");

delnodedll(f2);

displaydll(f2);

break;

case 3:

printf("\n");

printf("NO-CHANGES!\n");

displaydll(f2);

break;

default:

printf("Wrong-Number-Entered!\n");

break;

}

break;

case 3:

printf("\n");

printf("CIRCULAR-LINKED-LIST-PROGRAMES\n");

createcll(A,n);

printf("Existing Circular Linked List\n");

displaycll(f1);

printf("\n");

printf("Enter 1 to Add a new node\n");

printf("Enter 2 to Delete a node\n");

printf("Enter 3 for No-Changes and EXIT\n");

printf("Enter a Number: ");

scanf("%d",&y);

switch(y)

{

case 1:

printf("\n");

printf("Adding a New-Node!\n");

printf("Enter the data to be added: ");

scanf("%d",&t);

newnodecll(f1,t);

displaycll(f1);

break;

case 2:

printf("\n");

printf("Deleting a Node!\n");

delnodecll(f1);

displaycll(f1);

break;

case 3:

printf("\n");

printf("NO-CHANGES!");

displaycll(f1);

break;

default:

printf("\n");

printf("Wrong-Number-Entered!\n");

break;

}

break;

case 4:

printf("STACK-PROGRAMES\n");

printf("The Exsisting STACK\n");

createstack(A,n);

displaysll(f1);

printf("\n");

printf("Enter 1 to PUSH ITEM into the Stack\n");

printf("Enter 2 to POP ITEM out of the Stack\n");

printf("Enter 3 to EXIT");

printf("Enter a number: ");

scanf("%d",&y);

switch(y)

{

case 1:

printf("\n");

printf("PUSHING-ITEM-INTO-STACK\n");

printf("Enter the Data to be Pushed: ");

scanf("%d",&t);

push(t);

displaysll(f1);

break;

case 2:

printf("\n");

printf("POPING-ITEM-OUT-OF-THE-STACK\n");

pop(f1);

displaysll(f1);

break;

case 3:

printf("\n");

printf("NO-CHANGES\n");

displaysll(f1);

break;

default:

printf("\n");

printf("WRONG-NUMBER-ENTERED\n");

break;

}

break;

case 5:

printf("QUEUES-PROGRAMES\n");

printf("The Exsisting QUEUE\n");

createqll(A,n);

displayqll(f1);

printf("\n");

printf("Enter 1 to Add a new Element into the QUEUE\n");

printf("Enter 2 to Delete an Element from the QUEUE\n");

printf("Enter 3 for No-Changes and EXIT\n");

printf("Enter a number: ");

scanf("%d",&y);

switch(y)

{

case 1:

printf("\n");

printf("Adding a new element into the QUEUE\n");

printf("Enter the data to be added: ");

scanf("%d",&t);

newque(t);

displayqll(f1);

break;

case 2:

printf("\n");

printf("Deleting an Element from the Queue\n");

delque(f1);

displayqll(f1);

break;

case 3:

printf("\n");

printf("No-Changes\n");

displayqll(f1);

break;

default:

printf("\n");

printf("Wrong-Number-Entered\n");

break;

}

break;

default:

printf("Wrong-Number-Entered!\n");

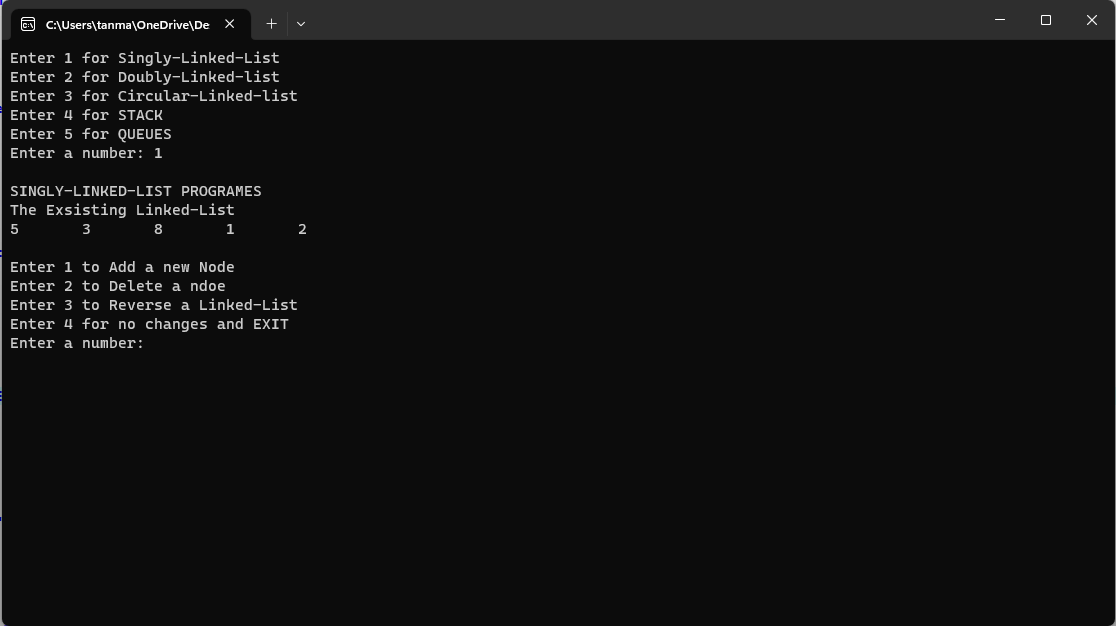
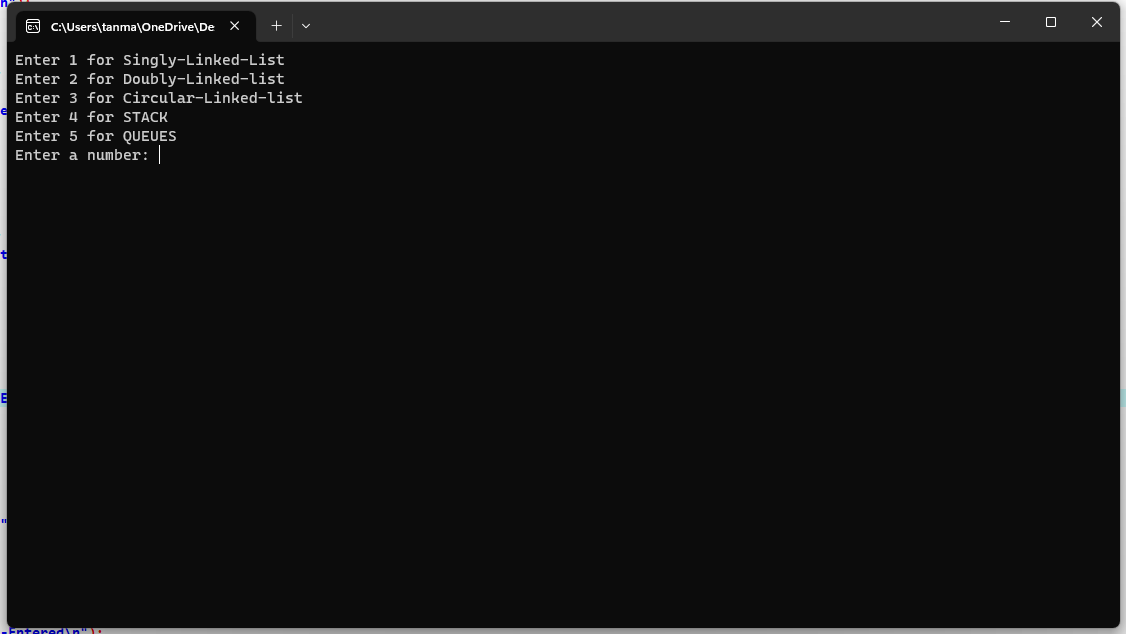
break;

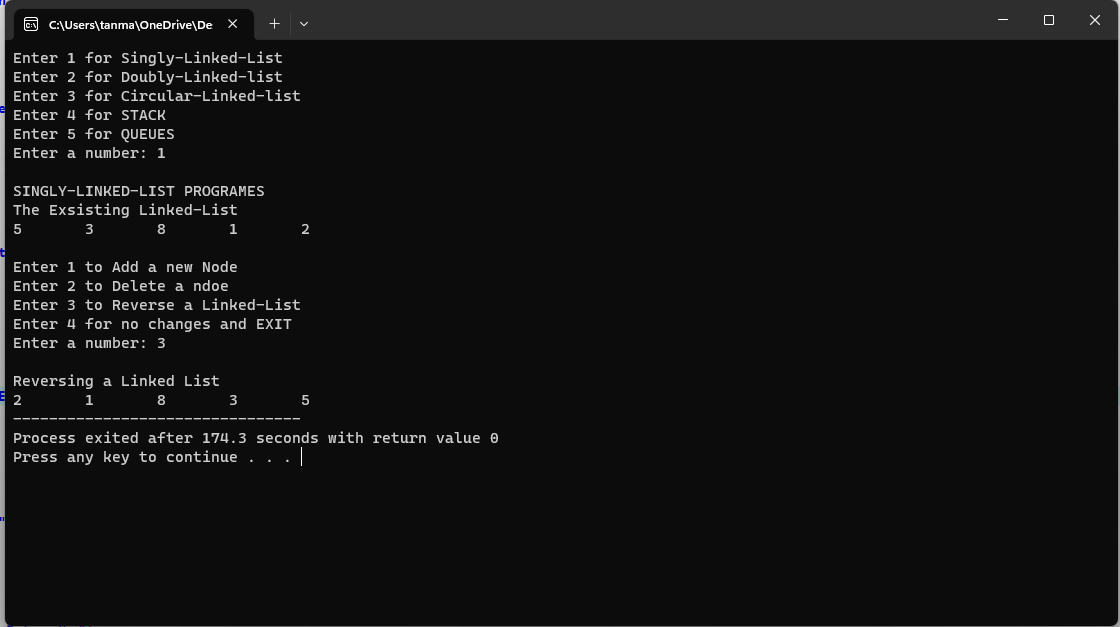
}

return 0;

}

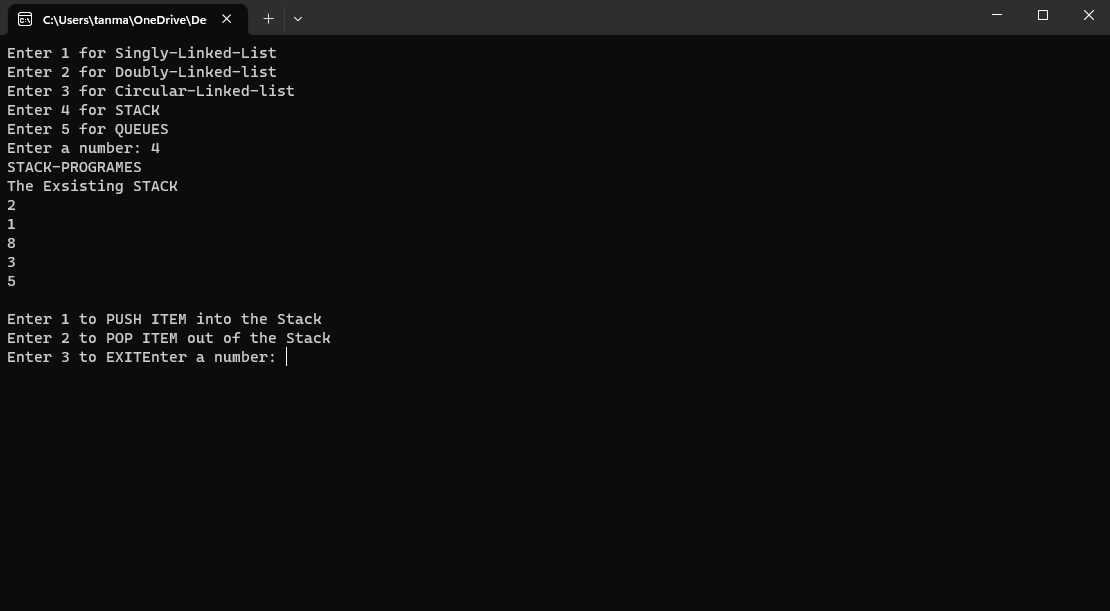
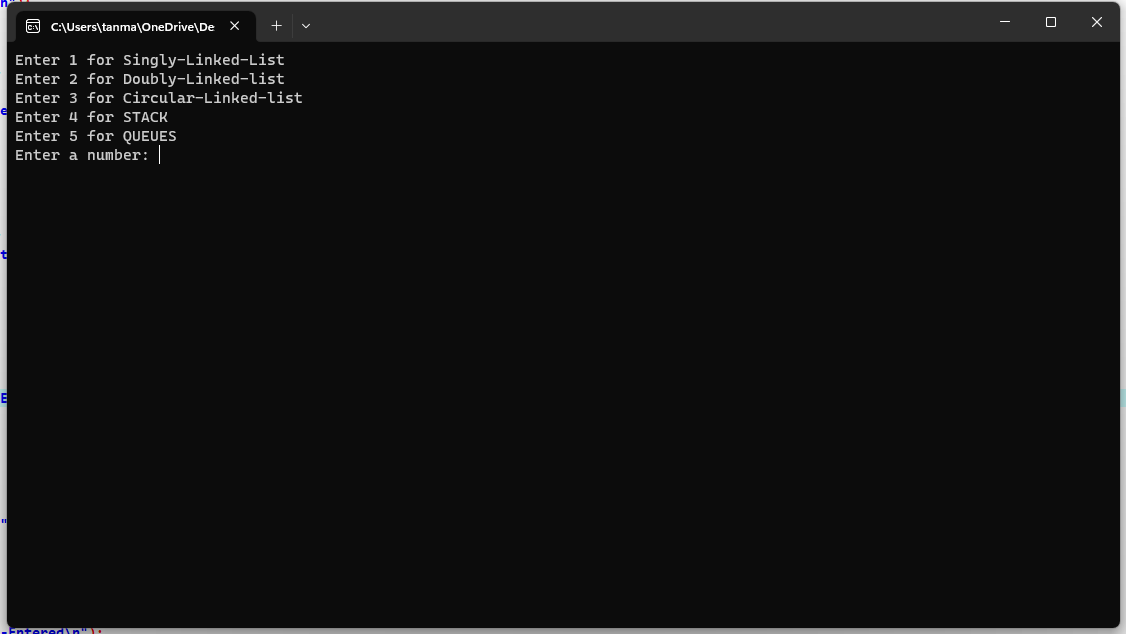
OUTPUT

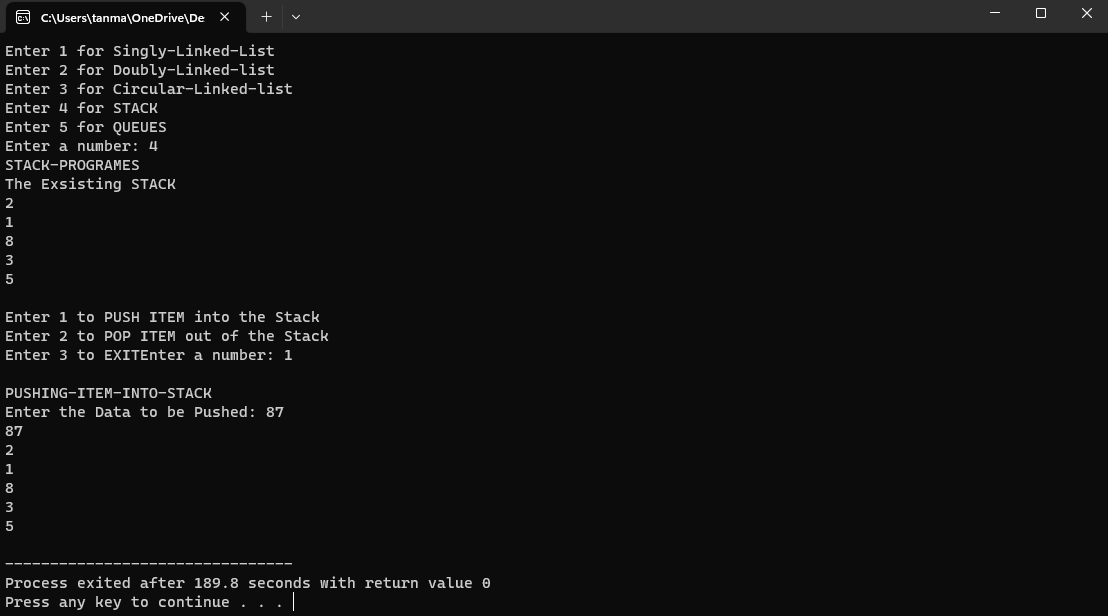
**EXAMPLE – 1**

****

We accessed the SINGLY-LINKED-LIST via the Menu Driven Program

**EXAMPLE – 2**

****



We accessed the STACK via the menu driven program

DS APPLICATIVE PROGRAMMING

DSA APPLICATIONS

Data Structures is one of the most widely used concepts across the IT Industry from changing of songs and making playlists to handling Big Data all are managed with the help of Data Structures.

We would be understanding how DSA is applied to solve/ simplify things with the help of an example application.

The Example used over here is **Changing of songs in a playlist**.

SPOTIFY WORKINGS

1. Spotify works on the basis of OOPS (Object Oriented Programming).
2. Spotify Creates Multiple Tables like Users table, Artists table, Albums table, Tracks table, Playlists table, Playlist Tracks table, Followers table, Likes table.
3. The Linking of the Data is done with the help of SQL concept of Keys – Primary Key & Foreign Key.
4. Each Table consists of some Foreign Keys which is used to link to the Primary Key database which helps in Relation between the two tables.
5. After the relations are formed the user gets song recommendations based on algorithms.

SOURCE CODE

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include<time.h>

struct dll

{

struct dll \*next;

struct dll \*prev;

char \*song;

} \*f1;

void createplaylist(char \*p[], int n)

{

struct dll \*t, \*last;

int i;

f1 = (struct dll \*)malloc(sizeof(struct dll));

f1->prev = NULL;

f1->next = NULL;

f1->song = p[0];

last = f1;

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

{

t = (struct dll \*)malloc(sizeof(struct dll));

t->next = NULL;

t->prev = last;

t->song = p[i];

last->next = t;

last = t;

}

}

void display(struct dll \*p)

{

while (p)

{

printf("%s\n", p->song);

p = p->next;

}

}

int main()

{

int n, x,i,y,t,m;

int sel;

char \*songin;

struct dll \*p,\*u;

time\_t k;

srand((unsigned) time(&k)); // Random Number Generator

n = 5;

char \*songs[] = {"1) No Promises by Cheat-Codes, Demi Lovato",

"2) Levels by AVICII",

"3) Pizza by Martin Garrix",

"4) Balam Pichkari by Pritam",

"5) Clarity by Zedd, Foxes"};

createplaylist(songs, n);

printf("The Current Playlist:\n");

display(f1);

printf("----------------------------------------------");

printf("\n");

printf("Enter 1 to play The first Song\n");

printf("Enter 2 to Select a song\n");

printf("Enter 3 to Select a Random Song\n");

printf("\n");

printf("Enter the number: ");

scanf("%d",&y);

printf("----------------------------------------------");

p=f1;

switch(y)

{

case 1:

printf("\n");

printf("Playing: %s",f1->song);

break;

case 2:

printf("\n");

printf("Enter the Song Number you want to play: ");

scanf("%d",&t);

printf("\n");

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

{

p=p->next;

}

printf("Playing: %s\n",p->song);

break;

case 3:

printf("\n");

m=rand()%6;

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

{

p=p->next;

}

printf("Playing: %s\n",p->song);

break;

}

printf("\n");

printf("\n");

printf("Enter 1 to go the Next song");

printf("\nEnter 2 to go to the Previous song\n");

printf("Enter:");

scanf("%d",&sel);

if(sel==1)

{

p=p->next;

if(p==NULL)

{

printf("\n");

printf("END OF PLATLIST!\n");

}

else

printf("\n");

printf("Now Playing: %s\n",p->song);

}

else

{

p=p->prev;

if(p==NULL)

{

printf("\n");

printf("END OF PLAYLIST\n");

}

else

printf("\n");

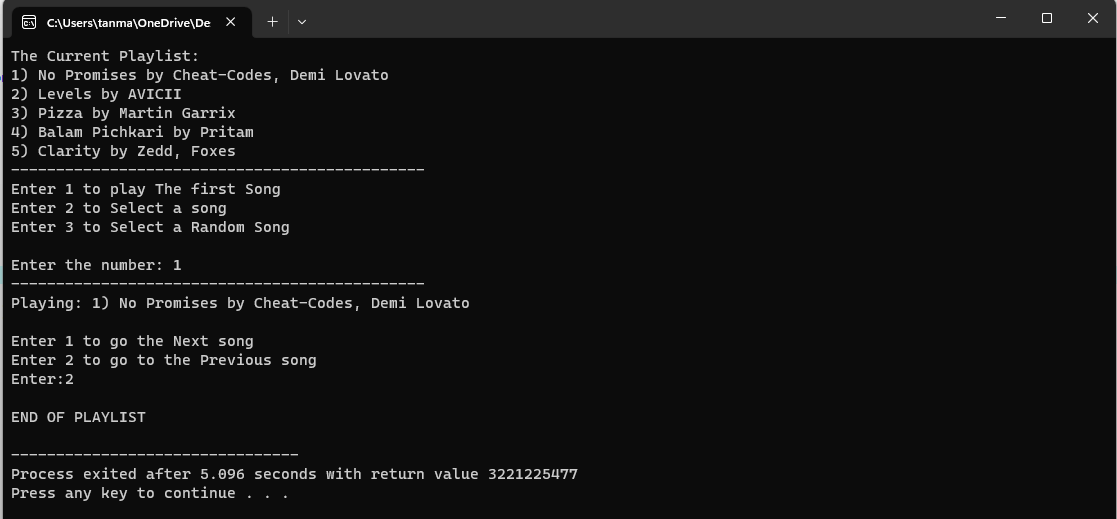
printf("Now Playing: %s\n",p->song);

}

printf("\n");

return 0;

}

 OUTPUT

MENU DRIVEN PROGRAM

BIBLIOGRAPHY

The materials/References which were used to better understand the project are:

1. Data Structures using C language – Reema Thareja
2. Class Notes
3. Spotify Research - <https://medium.com/towards-data-engineering/design-the-database-for-a-system-like-spotify-95ffd1fb5927>