**Sagar Institute of Science & Technology (SISTec), Bhopal (M.P.)**

****

**An ISO 9001:2008 Certified Institution**

**Approved by AICTE, New Delhi & Govt. of M.P.**

**Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)**

**Session: 2018-22**

**Department of COMPUTER SCIENCE & ENGINEERING**

**PROJECT REPORT:** All-in-One Data Structures

Submitted in partial fulfillment of the requirements

For the degree of

**BACHELOR OF TECHNOLOFY**

In

**COMPUTER SCIENCE & ENGINEERING**

Under the guidance of

**Mr. NARGISH GUPTA**

(Assistant Professor)

**Submitted by:**

Shivani Raichandani

0187EC181039

(3rd Semester, 2nd Year)

**Sagar Institute of Science & Technology (SISTec), Bhopal (M.P.)**

****

Department of Computer Science and Engineering

Session: 2018-22

**CERTIFICATE**

I hereby certify that, the work which is being presented in the B.Tech. Project Report, entitled as ***“All-in-One Data Structures”***, in partial fulfillment of the requirements for the award of ***Bachelor of Technology in Computer Science and*** ***Engineering*** and submitted to the ***Department of Computer Science and*** ***Engineering***. Sagar Institute of Science and Technology (SISTec), Bhopal (M.P.), is an authentic record of my own work carried out during the period of 3rd Semester i.e. (from July-2019 to November-2019) under the supervision of Mr. NARGISH GUPTA (Assistant Professor).

The content presented in this project has not been submitted by me for the reward of any other degree elsewhere.

Signature:

Shivani Raichandani

0187EC181039

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: October 16, 2019

Project Guide: HOD: Principal:

Table of Contents:

***Chapter: 1***

Introduction

* 1. Project Objective
  2. About Project

***Chapter: 2***

Software Requirements

***Chapter: 3***

UML Diagram

***Chapter: 4***

Code

***Chapter: 5***

Bibliography

**CHAPTER: 1**

***INTRODUCTION***

***Data Structures:***

Data Structures is a way to organize the data, into the memory. It is basically defined as the, “Data Organization, Management or Storage Technique, that enables efficient access and modification into data, in easy way.”

Data Structures

Primitive Data Structures

Non-Primitive Data Structures

Stack

Queue

Graphs

Trees

Non-linear Lists

Linear Lists

Arrays

Lists

Files

float

int

pointer

char

***Project Objective:***

The basic objective of this project is, to implement all the “Data Structures”, whether it may be – “Static or Dynamic Data Structures”, in a single program.

This will include:

1. Array
2. Stack
3. Queue
4. Linked List
5. Binary Search Tree

This project will enhance the logical thinking ability of an individual. It provides us the way, to organize a big code, at a single place.

***About Project:***

As the project includes all the data structures, thus the explanation of Data Structures is as follows:

**Array**:

An array is a non-primitive Data Structure. It is a collection of values of same datatype.

It is a static Data Structure, in which the memory is allocated at the compile time only.

**Queue**:

Queue is an “abstract Data Structure”, in which the entities in the collection are kept in order and the principal operation are the addition of the entities to “Rear” terminal, known as “Enqueue” and removal of the entities from “Front” terminal, known as “Dequeue”.

Queue is of following types:

* Simple Queue
* Circular Queue
* Priority Queue
* Deque

**Stack**:

Stack is a linear Data Structure, which follows the particular order in which the operations are performed. It is used to store the collections of objects.

Its basic principle is Last In First Out (LIFO). This means, the element inserted at very last position, will popped out firstly.

There are basic five applications of Stack, they are:

* Recursion
* Infix to Postfix Conversion
* Infix to Prefix Conversion
* Evaluation of Postfix Expression
* Tower of Hanoi

**Linked List**:

A Linked List is a dynamic Data Structure, where each element (called as Node), is made up of two items, the data and the address (reference or a pointer), which points to next Node.

**Binary Search Tree**:

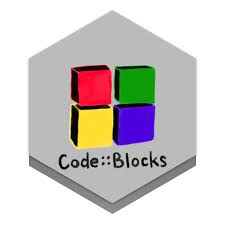
It is a tree or an Abstract Data Type (ADT), which is a node-based Binary Tree, whose main property is that, the left sub-tree of the node contains the keys lesser than the node’s key while the right sub-tree of the node contains the keys greater than the node’s key.

**CHAPTER: 2**

***SOFTWARE REQUIREMENTS***

***Software Requirements***:

For the coding purpose, I have used an Integrated Development Environment (IDE), named as: “Code::Blocks, version 17.12”.



It is an IDE, being developed for the WINDOWS and LINUX operating systems, and now also can be used by MAC OS users.

**Features**:

* Compiler: Code::Blocks support multiple compilers, including GCC, MinGW, Digital Mars, Microsoft Visual C++, etc.
* Code Editor: The IDE features “syntax highlighting” and “code folding”. Opened files are organized in tabs. The code editor supports font and font size selection and personalized syntax highlighting colors.
* Debugger: The Code::Blocks debugger has full breakpoint support. It also allows the user to debug their program by having access to the local function symbol and argument display, user-defined watches, call stack, custom memory dump and GNU Debugger interface.
* Project Files and Build System: Code::Blocks uses a custom build system, which stores its information in XML-based project files. It can optionally use external makefiles, which simplifies interfacing with projects using the GNU build system.

**CHAPTER: 3**

***UML DIAGRAM***

***UML Diagram***:

Run: OS

main()

[http://www.codeblocks.org/images/blank.png](http://www.codeblocks.org/)

Class Stack

Class Array

Class Linked List

insertion() push()

deletion() pop()

searching() display()

sorting() size\_of\_stack()

display() infix\_to\_prefix()

infix\_to\_postifix()

eval\_postfix()

Class Queue

enqueue() insertion()

dequeue() deletion()

searching() traversal()

sorting() reverse()

display()

Class Tree

insert()

delete()

display()

search()

max\_element()

The word UML stands for the Unified Modeling Language. It is a standard visual modeling language in the field of Software Engineering. It provides the standard way to visualize the design of the system to be deployed.

The above classes shown diagrammatically, represent the functions which are publicly accessible to the users, just to provide them an interface, so that, they are able to use this program and understand about the Data Structures more precisely and clearly.

***Deployment of the Code***:

The code basically is the representation of above diagram, and thus, whole of it is written using C++ language.

**CHAPTER: 4**

***CODE***

***CODE***:

using namespace std;

#include<bits/stdc++.h>

#include<algorithm>

#include <queue>

#include <stack>

#include <cstring>

#include<cstdlib>

#include<cstdio>

class Arr

{

int \*a,in;

public:

void insert(int n)

{

int x;

if(in==n)

{

cout<<"\nARRAY is FULL";

}

else{

cout<<"\nEnter the element";

cin>>x;

a[in]=x;

in++;

}

}

void del(int n)

{

int x;

cout<<"\nEnter the element";

cin>>x;

for (int i = 0; i < n; ++i)

{

if(a[i]==x)

{

for (int j = i; j < n; ++j)

{

a[j]=a[j+1];

}

in--;

break;

}

}

}

void search(int n)

{

int x;

cout<<"\nEnter the element";

cin>>x;

for (int i = 0; i < n; ++i)

{

if(a[i]==x)

{

cout<<"Position ="<<i;

cout<<"Address ="<<&a[i];

break;

}

}

}

void Sort(int n)

{

cout<<"\n Asscending order : \n";

sort(a,a+in);

for (int i = 0; i < in; ++i)

{

cout<<" "<<a[i];

}

cout<<"\n Decending order : \n";

sort(a,a+in,greater<int>());

for (int i = 0; i < in; ++i)

{

cout<<" "<<a[i];

}

}

void show()

{

int i=0;

while(a[i]!=0)

{

cout<<" "<<a[i];

i++;

}

}

void menu(int n)

{

a=new int [n+1];

in=0;

for (int i = 0; i < n+1; i++)

a[i] =0 ;

int x;

do

{

cout << "\033[1;31m";

cout<<"\t--------------------------------------\033[0m\n";

cout<<"\t\033[1;33m WELCOME TO ARRAY

\033[0m\n";

cout<<"\t\033[1;31m------------------------\033[0m\n\n";

cout << "\033[4;34m";

cout<<"1--insert an element\n";

cout<<"2--delete an element\n";

cout<<"3--search an element\n";

cout<<"4--sorting an element \n";

cout<<"5--Display\n";

cout<<"6--For Back Menu\n";

cout<<"7--Exit\033[0m\n";

cin>>x;

switch(x)

{

case 1:insert(n);cout<<"\nDone...";break;

case 2:del(n);cout<<"\nDone...";break;

case 3:search(n);break;

case 4:Sort(n);break;

case 5:show();break;

case 6:system("cls");return;break;

case 7:exit(0);

default:cout<<"\ninvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

};

class Stqueue

{

stack <int> s,s1;

public:

void menu()

{

int x;

do

{

cout << "\033[1;31m";

cout<<"\t--------------------------------------\033[0m\n";

cout<<"\t\033[1;33m WELCOME TO QUEUE (STACK)

\033[0m\n";

cout<<"\t\033[1;31m------------------------\033[0m\n\n";

cout << "\033[4;34m";

cout<<"1--insert an element\n";

cout<<"2--delete an element\n";

cout<<"3--Display\n";

cout<<"4--Back to menu\n";

cout<<"5--Exit\033[0m\n";

cin>>x;

switch(x)

{

case 1:insert();break;

case 2:s.pop();cout<<"\ndone...";break;

case 3:show();break;

case 4:return;break;

case 5:exit(0);

default:cout<<"\ninvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

void insert()

{

int x;

cout<<"\nEnter the element";

cin>>x;

while(!s.empty())

{

s1.push(s.top());

s.pop();

}

s.push(x);

while(!s1.empty())

{

s.push(s1.top());

s1.pop();

}

}

void show()

{

s1=s;

while(!s1.empty())

{

cout<<" "<<s1.top();

s1.pop();

}

}

};

class Dqueue

{

int \*a,rear,front;

public:

Dqueue(int n)

{

a=new int [n];

rear=front=-1;

}

void menu(int n)

{

int x;

do

{

cout << "\033[1;31m";

cout<<"\t------------------------------------\033[0m\n";

cout<<"\t\033[1;36m Welcome to Doubly Queue

\033[0m\n";

cout<<"\t\033[1;31m----------------------\033[0m\n\n";

cout << "\033[4;34m";

cout<<"1--insert an element in rear\n";

cout<<"2--insert an element in front\n";

cout<<"3--delete an element in front\n";

cout<<"4--delete an element in rear\n";

cout<<"5--search an element\n";

cout<<"6--sorting an element \n";

cout<<"7--Display\n";

cout<<"8--For Back Menu\n";

cout<<"9--Exit\033[0m\n";

cin>>x;

switch(x)

{

case 1:inrear(n);cout<<"\nDone...";break;

case 3:defront();cout<<"\nDone...";break;

case 5:search();break;

case 6:Sort(n);break;

case 7:show();break;

case 8:system("clear");return;break;

case 9:exit(0);

case 2:infront();cout<<"\nDone...";break;

case 4:derear();cout<<"\nDone...";break;

default:cout<<"\ninvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

void inrear(int n)

{

if (rear>n-1)

{

cout<<"\nqueue is FULL";

}

else

{

rear++;

cout<<"\nEnter the element";

cin>>a[rear];

if(front==-1)

front=0;

}

}

void infront()

{

if (front==0)

{

cout<<"\ninsertion cannot done";

}

else

{

front--;

cout<<"\nEnter the element";

cin>>a[front];

}

}

void defront()

{

if (front==-1||front>rear)

{

cout<<"\nQueue is empty";

}

else

{

int x=a[front];

front++;

}

}

void derear()

{

if(rear==-1||front>rear)

{

cout<<"\nQueue is empty";

}

else

{

int x=a[rear];

rear--;

}

}

void search()

{

int x;

cout<<"\nEnter the element";

cin>>x;

for (int i = front; i <=rear; i++)

{

if(a[i]==x)

{

cout<<"\nPosition = "<<i;

break;

}

}

}

void Sort(int n)

{

cout<<"\n Asscending order : \n";

sort(a,a+n);

for (int i = 0; i <=rear; i++)

{

cout<<" "<<a[i];

}

cout<<"\n Decending order : \n";

sort(a,a+n,greater<int>());

for (int i = 0; i <=rear; i++)

{

cout<<" "<<a[i];

}

}

void show()

{

for (int i = front; i <=rear; ++i)

{

cout<<" "<<a[i];

}

}

};

class Squeue

{

int \*a,rear,front;

public:

Squeue(int n)

{

a=new int [n];

rear=front=-1;

}

void menu(int n)

{

int x;

do

{

cout<<"\t\033[1;31m-----------------------\033[0m\n";

cout<<"\t\033[1;92m Welcome to Simple Queue

\033[0m\n";

cout<<"\t\033[1;31m----------------------\033[0m\n\n";

cout<<"\033[1;34m1--insert an element\n";

cout<<"2--delete an element\n";

cout<<"3--search an element\n";

cout<<"4--sorting an element \n";

cout<<"5--Display\n";

cout<<"6--For Back Menu\n";

cout<<"7--Exit\n\033[0m";

cin>>x;

switch(x)

{

case 1:insert(n);cout<<"\nDone...";break;

case 2:del();cout<<"\nDone...";break;

case 3:search();break;

case 4:Sort(n);break;

case 5:show();break;

case 6:system("cls");return;break;

case 7:exit(0);

default:cout<<"\ninvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

void insert(int n)

{

if (rear>n-1)

{

cout<<"\nqueue is FULL";

}

else

{

rear++;

cout<<"\nEnter the element";

cin>>a[rear];

if(front==-1)

front=0;

}

}

void del()

{

if (front==-1||front>rear)

{

cout<<"\nQueue is empty";

}

else

{

int x=a[front];

front++;

}

}

void search()

{

int x;

cout<<"\nEnter the element";

cin>>x;

for (int i = front; i <=rear; i++)

{

if(a[i]==x)

{

cout<<"\nPosition = "<<i;

break;

}

}

}

void Sort(int n)

{

cout<<"\n Asscending order : \n";

sort(a,a+n);

for (int i = 0; i <=rear; i++)

{

cout<<" "<<a[i];

}

cout<<"\n Decending order : \n";

sort(a,a+n,greater<int>());

for (int i = 0; i <=rear; i++)

{

cout<<" "<<a[i];

}

}

void show()

{

for (int i = front; i <=rear; ++i)

{

cout<<" "<<a[i];

}

}

};

class Cqueue

{

int \*a,rear,front;

public:

Cqueue(int n)

{

a=new int [n];

rear=front=-1;

}

void menu(int n)

{

int x;

do

{

cout<<"\t\033[1;31m------------------------\033[0m\n";

cout<<"\t\033[1;94mWelcome to Circular Queue

\033[0m\n";

cout<<"\t\033[1;31m----------------------\033[0m\n\n";

cout<<"\033[1;96m1--insert an element\n";

cout<<"2--delete an element\n";

cout<<"3--search an element\n";

cout<<"4--sorting an element \n";

cout<<"5--Display\n";

cout<<"6--For Back Menu\n";

cout<<"7--Exit\033[0m\n";

cin>>x;

switch(x)

{

case 1:insert(n);cout<<"\nDone...";break;

case 2:del(n);cout<<"\nDone...";break;

case 3:search(n);break;

case 4:Sort(n);break;

case 5:show(n);break;

case 6:system("cls");return;break;

case 7:exit(0);

default:cout<<"\ninvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

void insert(int n)

{

if ((front==0&&rear==n-1)||rear+1==front)

{

cout<<"\nqueue is FULL";

}

else

{

rear++;

rear=rear%n;

cout<<"\nEnter the element";

cin>>a[rear];

if(front==-1)

front=0;

}

}

void del(int n)

{

if (front==-1||front==rear+1)

{

cout<<"\nQueue is empty";

front=rear=-1;

}

else

{

int x=a[front];

front++;

front=front%n;

}

}

void search(int n)

{

int x;

cout<<"\nEnter the element";

cin>>x;

for (int i = front; i<rear; i=(i+1)%n)

{

if(a[i]==x)

{

cout<<"\nPosition = "<<i;

break;

}

}

}

void Sort(int n)

{

cout<<"\n Asscending order : \n";

sort(a,a+n);

for (int i = 0; i <=rear; i++)

{

cout<<" "<<a[i];

}

cout<<"\n Decending order : \n";

sort(a,a+n,greater<int>());

for (int i = 0; i <=rear; i++)

{

cout<<" "<<a[i];

}

}

void show(int n)

{

for (int i = front; i!=rear; i=(i+1)%(n+1))

{

cout<<" "<<a[i];

}

}

};

class Queues

{

public:

void menu(int n)

{

int x;

Squeue sq(n);

Cqueue cq(n);

Dqueue dq(n);

Stqueue stq;

do

{

cout<<"\t\033[1;31m------------------------\033[0m\n";

cout<<"\t\033[1;93m Welcome to Queue

\033[0m\n";

cout<<"\t\033[1;31m----------------------\033[0m\n\n";

cout<<"\033[1;34m1--Simple Queue\n";

cout<<"2--Circular Queue\n";

cout<<"3--Doubly Queue\n";

cout<<"4--Back to menu\n";

cout<<"5--Queue using stack\n";

cout<<"6--Exit\033[0m\n";

cin>>x;

switch(x)

{

case 1:system("cls");sq.menu(n);break;

case 2:system("cls");cq.menu(n);break;

case 3:system("cls");dq.menu(n);break;

case 4:system("cls");return;break;

case 6:exit(0);

case 5:system("cls");stq.menu();break;

default:cout<<"\ninvalid option";

}

system("cls");

}while(true);

}

};

class Intopost

{

public:

void start()

{

string p;

cout<<"\nenter the inflix expression ";

cin>>p;

convert(p);

}

void convert(string p)

{

stack <char> s;

s.push('(');

string pos;

for (int i = 0; i < p.size(); ++i)

{

if ((p[i]>='a'&&p[i]<='z')||(p[i]>='A'&&p[i]<='Z'))

{

pos+=p[i];

}

else

if(p[i]=='^'||p[i]=='\*'||p[i]=='/'||p[i]=='+'||p[i]=='-')

{

while(s.top()!='('&&imp(p[i])<=imp(s.top()))

{

char c=s.top();

s.pop();

pos=pos+c;

}

s.push(p[i]);

}

if(p[i]=='(')

s.push(p[i]);

if (p[i]==')')

{

while(s.top()!='(')

{

char c=s.top();

s.pop();

pos+=c;

}

if (s.top()=='(')

s.pop();

}

}

while(s.top()!='(')

{

char c=s.top();

s.pop();

pos=pos+c;

}

cout<<"postflix = "<<pos;

}

char imp(char x)

{

if(x=='^')

return (3);

else if (x=='\*'||x=='/')

return (2);

else if(x=='+'||x=='-')

return 1;

else

return 0;

}

};

class Stevl

{

stack <int> s;

public:

void start()

{

char p[20];

cout<<"\nenter the postflix expression ";

cin>>p;

for(int i=0;i<strlen(p);i++)

{

if(p[i]>='0'&&p[i]<'9')

s.push(p[i]-'0');

else

cal(p[i]);

}

}

void cal(char x)

{

int i,j;

i=s.top();

s.pop();

j=s.top();

switch(x)

{

case '^':s.top()=pow(j,i);break;

case '+':s.top()=j+i;break;

case '\*':s.top()=j\*i;break;

case '/':s.top()=j/i;break;

case '-':s.top()=j-i;break;

default :cout<<"\nno operator is found ";

}

}

void show()

{

cout<<"Answer = "<<s.top();

}

};

class Qstack

{

queue <int> q,q1;

public:

void menu()

{

int x;

do

{

cout<<"\t\033[1;31m-----------------------\033[0m\n";

cout<<"\t\033[1;91m Welcome to Stack(Queue)

\033[0m\n";

cout<<"\t\033[1;31m----------------------\033[0m\n\n";

cout<<"\033[1;34m1--insert an element\n";

cout<<"2--delete an element\n";

cout<<"3--Display\n";

cout<<"4--Back to menu\n";

cout<<"5--Exit\033[0m\n";

cin>>x;

switch(x)

{

case 1:insert();cout<<"\ndone...";break;

case 2:q.pop();cout<<"\ndone...";break;

case 3:show();break;

case 4:system("cls");return;break;

case 5:exit(0);

default:cout<<"\ninvalid option\n";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

void insert()

{

int a;

cout<<"\nEnter the element";

cin>>a;

q1.push(a);

while(!q.empty())

{

q1.push(q.front());

q.pop();

}

queue <int> q3=q;

q=q1;

q1=q3;

}

void show()

{

q1=q;

while(!q1.empty())

{

cout<<" "<<q1.front();

q1.pop();

}

}

};

class Intopre

{

stack <char> s;

public:

void start()

{

string p;

cout<<"\nenter the inflix expression ";

cin>>p;

reverse(p.begin(),p.end());

for (int i = 0; i < p.size(); ++i)

{

if(p[i]=='(')

{

p[i]=')';

i++;

}

else

if(p[i]==')')

{

p[i]='(';

i++;

}

}

convertpre(p);

}

void convertpre(string p)

{

s.push('(');

string pre;

for (int i = 0; i <p.size(); ++i)

{

if ((p[i]>='a'&&p[i]<='z')||(p[i]>='A'&&p[i]<='Z'))

{

pre+=p[i];

}

if(p[i]=='^'||p[i]=='\*'||p[i]=='/'||p[i]=='+'||p[i]=='-')

{

while(s.top()!='('&& imp(p[i])<=imp(s.top()))

{

char c=s.top();

s.pop();

pre+=c;

}

s.push(p[i]);

}

if(p[i]=='(')

s.push(p[i]);

if (p[i]==')')

{

while(s.top()!='(')

{

char c=s.top();

s.pop();

pre+=c;

}

if (s.top()=='(')

s.pop();

}

}

while(s.top()!='(')

{

char c=s.top();

s.pop();

pre+=c;

}

reverse(pre.begin(),pre.end());

cout<<"preflix = "<<pre;

}

char imp(char x)

{

if(x=='^')

return (3);

else if (x=='\*'||x=='/')

return (2);

else if(x=='+'||x=='-')

return 1;

else

return 0;

}

};

class Stack

{

stack <int> s;

public:

void menu()

{

int x;

Qstack qs;

Stevl st;

Intopost pos;

Intopre pre;

do

{

cout<<"\t\033[1;31m------------------------\033[0m\n";

cout<<"\t\033[1;35mWelcome to Stack \033[0m\n";

cout<<"\t\033[1;31m----------------------\033[0m\n\n";

cout<<"\033[1;34m1--insert an element\n";

cout<<"2--delete an element\n";

cout<<"3--Display\n";

cout<<"4--Size of Stack\n";

cout<<"5--Stack using Queue \n";

cout<<"6--infix to postfix\n";

cout<<"7--infix to prefix\n";

cout<<"8--Evaluation of postfix\n";

cout<<"9--Back to menu\n";

cout<<"10--Exit\033[0m\n";

cin>>x;

switch(x)

{

case 1:insert();cout<<"\ndone...";break;

case 2:s.pop();cout<<"\ndone...";break;

case 3:show();break;

case 9:return;break;

case 10:exit(0);

case 4:cout<<"\nSize = "<<s.size();break;

case 5:system("cls");qs.menu();break;

case 6:pos.start();break;

case 7:pre.start();break;

case 8:st.start();st.show();break;

default:cout<<"\ninvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

void insert()

{

int a;

cout<<"\nEnter the element";

cin>>a;

s.push(a);

}

void show()

{

stack <int> s1=s;

while(!s1.empty())

{

cout<<" "<<s1.top();

s1.pop();

}

}

};

struct node

{

int data;

struct node \*next;

};

struct link

{

int data;

struct link \*com,\*pre;

};

class Dlinked

{

link \*start,\*back;

public :

Dlinked()

{

back=start=NULL;

}

void inend(int x)

{

struct link \*d=new struct link;

d->data=x;

d->pre=NULL;

if(start==NULL)

{

cout<<"\nlinklist was empty ";

start=d;

back=d;

cout<<"\nlinklist was created ";

}

else

{

back->com=d;

d->com=NULL;

d->pre=back;

back=d;

}

}

void show()

{

cout<<"\t\*\*\*\*\*Linked list \*\*\*\*\*\n";

struct link \*reuse;

reuse=start;

do

{

cout<<reuse->data<<endl;

reuse=reuse->com;

} while(reuse!=NULL);

}

void infront(int x)

{

struct link \*d=new struct link;

d->data=x;

d->com=NULL;

if(start==NULL)

{

cout<<"\nlinklist was empty ";

start=d;

back=d;

cout<<"\nlinklist was created ";

}

else

{

d->com=start;

start->pre=d;

d->pre=NULL;

start=d;

}

}

void inpos(int x,int pos)

{

struct link \*d=new struct link;

struct link \*reuse,\*reuse1;

int i=1;

d->data=x;

reuse=start;

while(reuse->com!=NULL&&i<pos)

{

i++;

reuse1=reuse;

reuse=reuse->com;

}

reuse1->com=d;

d->pre=reuse1;

d->com=reuse;

reuse->pre=d;

}

void deend()

{

struct link \*reuse,\*reuse1;

reuse=start;

do

{

reuse1=reuse;

reuse=reuse->com;

}while(reuse->com!=NULL);

reuse1->com=NULL;

back=reuse1;

delete reuse;

}

void defront()

{

struct link \*reuse;

reuse=start;

start=start->com;

start->pre=NULL;

delete reuse;

}

void depos(int pos)

{

struct link \*reuse,\*reuse1;

int i=1;

reuse=start;

while(reuse->com!=NULL&&i<pos)

{

i++;

reuse1=reuse;

reuse=reuse->com;

}

reuse1->com=reuse->com;

reuse1=reuse->com;

reuse1->pre=reuse->pre;

delete reuse;

}

void search(int x)

{

struct link \*reuse,\*reuse1;

reuse=start;

int pos=0;

while(reuse->com!=NULL)

{

pos++;

if(reuse->data==x)

{

cout<<"\nPosition = "<<pos;

cout<<"\nAddress = "<<reuse;

break;

}

reuse1=reuse;

reuse=reuse->com;

}

if(reuse->data!=x)

cout<<"\nNot found ";

}

void tran()

{

struct link \*reuse,\*reuse1;

reuse=start;

do

{

reuse1=reuse;

reuse=reuse->com;

}while(reuse->com!=NULL);

cout<<"\ntranverse : \n Last = "<<reuse<<"\n 2nd last = "<<reuse1;

}

void rev()

{

struct link \*reuse;

reuse=back;

cout<<"\t\*\*\*\*\*Linked list \*\*\*\*\*\n";

while(reuse!=NULL)

{

cout<<reuse->data<<endl;

reuse=reuse->pre;

}

}

void menu()

{

int x,a,pos;

do

{

cout<<"\t\033[1;31m----------------------\033[0m\n";

cout<<"\t\033[1;92mWelcome to Doubly Linked

Lists \033[0m\n";

cout<<"\t\033[1;31m---------------------\033[0m\n\n";

cout<<"\033[1;34m1--insert at end \n";

cout<<"2--insert at front \n";

cout<<"3--insert at position \n";

cout<<"4--delete at end \n";

cout<<"5--delete at front \n";

cout<<"6--delete at position \n";

cout<<"7--Search the element \n";

cout<<"8--tranverse \n";

cout<<"9--Reverse Display \n";

cout<<"10--Display \n";

cout<<"11--Exit\n";

cout<<"12--Back to menu\033[0m\n";

cin>>a;

switch(a)

{

case 1:

cout<<"enter the data :";

cin>>x;

inend(x);break;

case 2:

cout<<"enter the data :";

cin>>x;

infront(x);break;

case 3:

cout<<"enter the data :";

cin>>x;

cout<<"\nenter the position :";

cin>>pos;

inpos(x,pos);break;

case 4:

deend();break;

case 5:

defront();break;

case 6:

cout<<"\nenter the position :";

cin>>pos;

depos(pos);break;

case 7:

cout<<"enter the element u want to search :";

cin>>x;

search(x);break;

case 8:

tran();break;

case 9:

rev();break;

case 10:

show();break;

case 11:exit(0);

case 12:system("cls");return;

default:cout<<"\n\tinvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

};

class Cdlinked

{

link \*start,\*back;

public :

Cdlinked()

{

back=start=NULL;

}

void inend(int x)

{

struct link \*d=new struct link;

d->data=x;

d->pre=NULL;

if(start==NULL)

{

cout<<"\nlinklist was empty ";

start=d;

back=d;

cout<<"\nlinklist was created ";

}

else

{

back->com=d;

d->com=start;

d->pre=back;

back=d;

start->pre=back;

}

}

void show()

{

cout<<"\t\*\*\*\*\*Linked list \*\*\*\*\*\n";

struct link \*reuse;

reuse=start;

while(reuse->com!=start)

{

cout<<reuse->data<<endl;

reuse=reuse->com;

}

cout<<reuse->data<<endl;

}

void infront(int x)

{

struct link \*d=new struct link;

d->data=x;

d->com=NULL;

if(start==NULL)

{

cout<<"\nlinklist was empty ";

start=d;

back=d;

cout<<"\nlinklist was created ";

}

else

{

d->com=start;

start->pre=d;

d->pre=back;

start=d;

}

}

void inpos(int x,int pos)

{

struct link \*d=new struct link;

struct link \*reuse,\*reuse1;

int i=1;

d->data=x;

reuse=start;

while(reuse->com!=start&&i<pos)

{

i++;

reuse1=reuse;

reuse=reuse->com;

}

reuse1->com=d;

d->pre=reuse1;

d->com=reuse;

reuse->pre=d;

}

void deend()

{

struct link \*reuse,\*reuse1;

reuse=start;

do

{

reuse1=reuse;

reuse=reuse->com;

}while(reuse->com!=start);

reuse1->com=start;

back=reuse1;

start->pre=back;

delete reuse;

}

void defront()

{

struct link \*reuse;

reuse=start;

start=start->com;

start->pre=back;

delete reuse;

}

void depos(int pos)

{

struct link \*reuse,\*reuse1;

int i=1;

reuse=start;

while(reuse->com!=start&&i<pos)

{

i++;

reuse1=reuse;

reuse=reuse->com;

}

reuse1->com=reuse->com;

reuse1=reuse->com;

reuse1->pre=reuse->pre;

delete reuse;

}

void search(int x)

{

struct link \*reuse,\*reuse1;

reuse=start;

int pos=0;

while(reuse->com!=start)

{

pos++;

if(reuse->data==x)

{

cout<<"\nPosition = "<<pos;

cout<<"\nAddress = "<<reuse;

break;

}

reuse1=reuse;

reuse=reuse->com;

}

if (back->data==x)

{

cout<<"\nPosition = "<<pos;

cout<<"\nAddress = "<<reuse;

}

if(reuse->data!=x)

cout<<" \nelement not found\n";

}

void menu()

{

int x,a,pos;

do

{

cout<<"\t\033[1;31m-----------------------\033[0m\n";

cout<<"\t\033[1;92m Welcome to Circular Doubly

Linked Lists \033[0m\n";

cout<<"\t\033[1;31m---------------------\033[0m\n\n";

cout<<"\033[1;34m1--insert at end \n";

cout<<"2--insert at front \n";

cout<<"3--insert at position \n";

cout<<"4--delete at end \n";

cout<<"5--delete at front \n";

cout<<"6--delete at position \n";

cout<<"7--Search the element \n";

cout<<"8--Display \n";

cout<<"9--Exit\n";

cout<<"10--Back to menu\033[0m\n";

cin>>a;

switch(a)

{

case 1:

cout<<"enter the data :";

cin>>x;

inend(x);break;

case 2:

cout<<"enter the data :";

cin>>x;

infront(x);break;

case 3:

cout<<"enter the data :";

cin>>x;

cout<<"\nenter the position :";

cin>>pos;

inpos(x,pos);break;

case 4:

deend();break;

case 5:

defront();break;

case 6:

cout<<"\nenter the position :";

cin>>pos;

depos(pos);break;

case 7:

cout<<"enter the element u want to search :";

cin>>x;

search(x);break;

case 8:

show();break;

case 9:exit(0);

case 10:system("cls");return;

default:cout<<"\n\tinvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

};

class Clinked

{

node \*head,\*tail;

public :

Clinked()

{

tail=head=NULL;

}

void inend(int x)

{

struct node \*p=new struct node;

p->data=x;

p->next=NULL;

if(head==NULL)

{

cout<<"\nlinklist was empty ";

head=p;

tail=p;

cout<<"\nlinklist was created ";

}

else

{

tail->next=p;

p->next=head;

tail=p;

}

}

void show()

{

cout<<"\t\*\*\*\*\*Linked list \*\*\*\*\*\n";

struct node \*tem;

tem=head;

do

{

cout<<tem->data<<endl;

tem=tem->next;

} while(tem->next!=head);

cout<<tem->data<<endl;

}

void infront(int x)

{

struct node \*p=new struct node;

p->data=x;

p->next=NULL;

if(head==NULL)

{

cout<<"\nlinklist was empty ";

head=p;

tail=p;

cout<<"\nlinklist was created ";

}

else{

p->next=head;

tail->next=p;

head=p;

}}

void inpos(int x,int pos)

{

struct node \*p=new struct node;

struct node \*tem,\*tem1;

int i=1;

p->data=x;

tem=head;

do

{

i++;

tem1=tem;

tem=tem->next;

} while(tem1->next!=head&&i<pos);

tem1->next=p;

p->next=tem;

}

void deend()

{

struct node \*tem,\*tem1;

tem=head;

do

{

tem1=tem;

tem=tem->next;

}while(tem->next!=head);

tem1->next=head;

tail=tem1;

delete tem;

}

void defront()

{

struct node \*tem;

tem=head;

head=head->next;

tail->next=head;

delete tem;

}

void depos(int pos)

{

struct node \*tem,\*tem1;

int i=1;

tem=head;

do

{

i++;

tem1=tem;

tem=tem->next;

}while(tem->next!=head);

tem1->next=tem->next;

delete tem;

}

void search(int x)

{

struct node \*tem,\*tem1;

tem=head;

int pos=0;

do

{

pos++;

if(tem->data==x)

{

cout<<"\nPosition = "<<pos;

cout<<"\nAddress = "<<tem;

break;

}

tem1=tem;

tem=tem->next;

}while(tem1->next!=head);

if(tem->data!=x)

cout<<"\nNot found ";

}

void tran()

{

struct node \*tem,\*tem1;

tem=head;

do

{

tem1=tem;

tem=tem->next;

}while(tem1->next!=head);

cout<<"\ntranverse : \n Last = "<<tem<<"\n 2nd last = "<<tem1;

}

void menu()

{

int x,a,pos;

do

{

cout<<"\t\033[1;31m-----------------\033[0m\n";

cout<<"\t\033[1;92m Welcome to Circular Linked

Lists \033[0m\n";

cout<<"\t\033[1;31m-------------------\033[0m\n\n";

cout<<"\033[1;34m1--insert at end \n";

cout<<"2--insert at front \n";

cout<<"3--insert at position \n";

cout<<"4--delete at end \n";

cout<<"5--delete at front \n";

cout<<"6--delete at position \n";

cout<<"7--Search the element \n";

cout<<"8--traverse \n";

cout<<"9--Display \n";

cout<<"10--Exit\n";

cout<<"11--Back to menu\033[0m\n";

cin>>a;

switch(a)

{

case 1:

cout<<"enter the data :";

cin>>x;

inend(x);break;

case 2:

cout<<"enter the data :";

cin>>x;

infront(x);break;

case 3:

cout<<"enter the data :";

cin>>x;

cout<<"\nenter the position :";

cin>>pos;

inpos(x,pos);break;

case 4:

deend();break;

case 5:

defront();break;

case 6:

cout<<"\nenter the position :";

cin>>pos;

depos(pos);break;

case 7:

cout<<"enter the element u want to search :";

cin>>x;

search(x);break;

case 8:

tran();break;

case 9:

show();break;

case 10:

exit(0);

case 11:system("cls");return;break;

default:cout<<"\n\tinvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

};

class Slinked

{

node \*head,\*tail;

public :

Slinked()

{

tail=head=NULL;

}

void inend(int x)

{

struct node \*p=new struct node;

p->data=x;

p->next=NULL;

if(head==NULL)

{

cout<<"\nLinked list does not exist.";

head=p;

tail=p;

cout<<"\nLinked list is Created!!";

}

else

{

tail->next=p;

tail=p;

tail->next=head;

}

}

void show()

{

cout<<"\t\*\*\*\*\*Linked list \*\*\*\*\*\n";

struct node \*tem;

tem=head;

while(tem!=NULL)

{

cout<<tem->data<<endl;

tem=tem->next;

}

}

void infront(int x)

{

struct node \*p=new struct node;

p->data=x;

if(head==NULL)

{

cout<<"\nLinked list is empty... Try again... ";

head=p;

tail=p;

cout<<"\nLinked list Created!! ";

}

else

{

p->next=head;

head=p;

tail->next=head;

}

}

void inpos(int x,int pos)

{

struct node \*p=new struct node;

struct node \*tem,\*tem1;

int i=1;

p->data=x;

tem=head;

while(tem!=NULL&&i<pos)

{

i++;

tem1=tem;

tem=tem->next;

}

tem1->next=p;

p->next=tem;

}

void deend()

{

struct node \*tem,\*tem1;

tem=head;

while(tem->next!=NULL)

{

tem1=tem;

tem=tem->next;

}

tail=tem1;

tail->next=head;

delete tem;

}

void defront()

{

struct node \*tem;

tem=head;

head=head->next;

tail->next=head;

delete tem;

}

void depos(int pos)

{

struct node \*tem,\*tem1;

int i=1;

tem=head;

while(tem!=NULL&&i<pos)

{

i++;

tem1=tem;

tem=tem->next;

}

tem1->next=tem->next;

delete tem;

}

void search(int x)

{

struct node \*tem;

tem=head;

int pos=0;

while(tem!=NULL)

{

pos++;

if(tem->data==x)

{

cout<<"\nPosition = "<<pos;

cout<<"\nAddress = "<<tem;

break;

}

tem=tem->next;

}

if(tem->data!=x)

cout<<"\nNot found ";

}

void tran()

{

struct node \*tem,\*tem1;

tem=head;

while(tem->next!=NULL)

{

tem1=tem;

tem=tem->next;

}

cout<<"\ntranverse : \n Last = "<<tem<<"\n 2nd last = "<<tem1;

}

void rev()

{ cout<<"\t\*\*\*\*\*Linked list \*\*\*\*\*\n";

reverse(head);}

void reverse(node \*tem)

{

if(tem->next!=head)

reverse(tem->next);

cout<<tem->data<<endl;

}

void menu()

{

int x,a,pos;

do

{

cout<<"\t\033[1;31m-------------------\033[0m\n";

cout<<"\t\033[1;92m Welcome to Simple Linked

Lists \033[0m\n";

cout<<"\t\033[1;31m-------------------\033[0m\n\n";

cout<<"\033[1;34m1--insert at end \n";

cout<<"2--insert at front \n";

cout<<"3--insert at position \n";

cout<<"4--delete at end \n";

cout<<"5--delete at front \n";

cout<<"6--delete at position \n";

cout<<"7--Search the element \n";

cout<<"8--Traverse the Linked List \n";

cout<<"9--Reverse Display \n";

cout<<"10--Display \n";

cout<<"11--Back to menu\n";

cout<<"12--Exit\033[0m\n";

cin>>a;

switch(a)

{

case 1:

cout<<"Enter the data :";

cin>>x;

inend(x);break;

case 2:

cout<<"Enter the data :";

cin>>x;

infront(x);break;

case 3:

cout<<"Enter the data :";

cin>>x;

cout<<"\nEnter the position :";

cin>>pos;

inpos(x,pos);break;

case 4:

deend();break;

case 5:

defront();break;

case 6:

cout<<"\nEnter the position :";

cin>>pos;

depos(pos);break;

case 7:

cout<<"Enter the element to search :";

cin>>x;

search(x);break;

case 8:

tran();break;

case 9:

rev();break;

case 10:

show();break;

case 11:system("cls");return;break;

case 12:exit(0);

default:cout<<"\n\tInvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

};

class Linked

{

public:

void menu()

{

int x;

Slinked sl;

Clinked cl;

Dlinked dl;

Cdlinked cdl;

do

{

cout<<"\t\033[1;31m----------------------------\033[0m\n";

cout<<"\t\033[1;92m Welcome to Linked Lists

\033[0m\n";

cout<<"\t\033[1;31m---------------------------\033[0m\n\n";

cout<<"\033[1;34m1--Simple LinkedLists\n";

cout<<"2--Circular LinkedLists\n";

cout<<"3--Doubly LinkedLists\n";

cout<<"4--Circular Doubly LinkedLists\n";

cout<<"5--Back to menu\n";

cout<<"6--Exit\033[0m\n";

cin>>x;

switch(x)

{

case 1:system("cls");sl.menu();break;

case 2:system("cls");cl.menu();break;

case 3:system("cls");dl.menu();break;

case 4:system("cls");cdl.menu();break;

case 5:system("cls");return ;

case 6:exit(0);

default:cout<<"\ninvalid option";

}

system("cls");

}while(true);

}

};

struct node1

{

int data;

struct node1 \*left,\*right;

};

class tree

{

node1 \*root;

public:

tree()

{

root=NULL;

}

void create()

{

struct node1 \*p= new struct node1;

int x;

cout<<"\nenter data";

cin>>x;

p->data=x;

p->left=p->right=NULL;

if (root==NULL)

{

root=p;

}

else

insert(root,p);

}

void insert(struct node1 \*root,struct node1 \*p)

{

if(root->data>p->data)

{

if(root->left!=NULL)

insert(root->left,p);

else

root->left=p;

}

if(root->data<p->data)

{

if(root->right!=NULL)

insert(root->right,p);

else

root->right=p;

}

}

void search(struct node1 \*tem,int x)

{

static int i=0;

if(tem->data>=x)

{

i++;

if(tem->data!=x)

search(tem->left,x);

else

{

cout<<"\naddress = "<<tem;

cout<<"\nPosition = "<<i;

i=0;

}

}

if(tem->data<x)

{

i++;

if(tem->data!=x)

search(tem->right,x);

}

}

struct node1 \*del(struct node1 \*tem,int x)

{

if(tem->data>x)

tem->left=del(tem->left,x);

else

if(tem->data<x)

tem->right=del(tem->right,x);

else

{

if(tem->left==NULL&&tem->right==NULL)

{

delete tem;

tem=NULL;

return tem;

}

else

if(tem->left==NULL)

{

struct node1 \*tem1=tem;

tem1=tem->right;

tem=NULL;

delete tem;

return tem1;

}

else

if (tem->right==NULL)

{

struct node1 \*tem1=tem;

tem1=tem->left;

tem=NULL;

delete tem;

return tem1;

}

else

{

struct node1 \*tem1;

tem1=lmost(tem->right);

tem->data=tem1->data;

tem->right=del(tem->right,tem1->data);

}

}

return tem;

}

struct node1 \*lmost(struct node1 \*tem)

{

while(tem->left!=NULL)

{

tem=tem->left;

}

return tem;

}

void show(int x)

{

switch(x)

{

case 1:inorder(root);break;

case 2:preorder(root);break;

case 3:postorder(root);break;

}

}

void inorder(struct node1 \*tem)

{

if(tem!=NULL)

{

inorder(tem->left);

cout<<tem->data<<" ";

inorder(tem->right);

}

}

void preorder(struct node1 \*tem)

{

if(tem!=NULL)

{

cout<<tem->data<<" ";

inorder(tem->left);

inorder(tem->right);

}

}

void postorder(struct node1 \*tem)

{

if(tem!=NULL)

{

inorder(tem->left);

inorder(tem->right);

cout<<tem->data<<" ";

}

}

void max(struct node1 \*tem)

{

while(tem->right!=NULL)

{

tem=tem->right;

}

cout<<"\n max = "<<tem->data;

}

void menu()

{

int a,x;

do

{

cout<<"\t\033[1;31m---------------------------------\033[0m\n";

cout<<"\t\033[1;92m Welcome to Binary Search Tree

\033[0m\n";

cout<<"\t\033[1;31m---------------------------------\033[0m\n\n";

cout<<"\n\033[1;34m1--insert\n";

cout<<"2--display\n";

cout<<"3-search \n";

cout<<"4--delete\n";

cout<<"5--Max element\n";

cout<<"6--Back to menu\n";

cout<<"7--exit\033[0m\n";

cin>>a;

switch(a)

{

case 1:create();break;

case 2:

cout<<"1--inorder\n";

cout<<"2--preorder\n";

cout<<"3--postorder\n";

cin>>x;

show(x);break;

case 3: cout<<"\nenter the value = ";

cin>>x;

search(root,x);break;

case 4:

cout<<"\nenter the value = ";

cin>>x;

del(root,x);break;

case 5:max(root);break;

case 6:system("cls");return;break;

case 7:exit(0);

default:cout<<"\n\tinvalid option";

}

cout<<endl;

system("pause");

system("cls");

}while(true);

}

};

int main()

{

int x,n;

Stack s;

Queues q;

Linked l;

tree t;

Arr a;

do

{

cout<<"\t-------------------------------------------------------\n";

cout<<"\t WELCOME TO DATA STRUCTURE \n";

cout<<"\t---------------------------------------------------\n\n";

cout<<"\t1--Array\n";

cout<<"\t2--Queue\n";

cout<<"\t3--Stack\n";

cout<<"\t4--LinkedLists\n";

cout<<"\t5--Tree\n";

cout<<"\t6--Exit\n";

cin>>x;

switch(x)

{

case 1: cout<<"\nPlease Enter the size of ARRAY : ";

cin>>n;

system("cls");

a.menu(n);break;

case 2:

cout<<"\nPlease Enter the size of Queue : ";

cin>>n;

system("cls");

q.menu(n);break;

case 3:system("cls");s.menu();break;

case 4:system("cls");l.menu();break;

case 5:system("cls");t.menu();break;

case 6:exit(0);

default :cout<<"\n invalid option\n";

}

}while(true);

}

\*\*\*\*\*\*\*\*\*\*\*\*

**CHAPTER: 5**

***BIBLIOGRAPHY***

***Bibliography***:

Sources and References:

* Data Structures (Schaum’s Series)
* Data Structures using C and C++ (By Yedidyah Langsam)
* Google (Geeks for Geeks)