**Ex: Write a program to reed student name and three Degrees and find the average using struct.**

**Sol:**

# include < iostrem.h>

Struct student { char name[10];

Int d1, d2, d3;

Float Av;

};

Main ( )

{ Student recst;

Cout <<" enter the name ";

Cin >> recst. Name;

Cout<< "enter the first deg";

Cin >> recst. d1;

Cout<< "enter the second deg";

Cin >> recst.d2;

Cout<<" enter the third deg";

Cin>> recst.d3;

recst .Av=(recst.d1+recst.d2+recst.d3)/3.0;

Cout<<"Av="<< recst .Av;

}

**Ex: writ a program to find average of n students has every one (8) degrees, using struct?**

**Sol:**

# include <iostrem.h>

Struct student

{

Char name[10];

int d[8] ;

float AV;

};

main ( )

{ Student A[n]; int n, i, j;

cout <<" enter n"; cin>>n;

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

{

Cout<<"enter the name";

Cin>>A[i]. name

Sum=0

for (j=1;j<=8 ; j++)

{ Cout<<"enter deg ";

Cin>> A[i].d[j];

Sum= sum + A[i] .d[j];

}

A[i].Av= sum / 8.0;

}

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

Cout<<A[i].Av;

}

**Program of stack**

# include <iostream .h>

# defined max 3

int stack [ max];

int top = 0 ;

**Void push (int a)**

{if ( top == 3)

Cout <<" stack is full";

else

{top ++ ;

Stack [top] =a;

}

}

**Void pop ( )**

{int x ;

if (top == 0) cout <<" stack is empty";

else

{x = stack [top];

top -- ;

Cout <<" deleted" << x;

}

}

Main ( )

Push (1);

Push (2);

Push (3);

Push (4);

Pop () ;

Pop () ;

Pop () ;

Pop () ;

}

# Example Program Of Infix To Prefix

#include<stdio.h>

#include<math.h>

#define max 100

int stack[max];

int top=-1;

void push(int a)

{

if(top==99)

cout<<"\nStack is Full";

else

{ top++;

stack[top]=a;

}

}

int pop()

{ int x;

if(top==-1)

cout<<"\nStack is Empty";

else

{ x=stack[top];

top--;

return x;

}

return 0;

}

void main()

{

char infix[100];

//char prefix[100];

//int psol;

//void presol(char []);

void infixpre(char[]);

clrscr();

cout<<"Enter the infixexpession :";

gets(infix);

infixpre(infix);

//getch();

// cout<<"Enter a prefix expression:" ;gets(prefix);

//presol(prefix);

}

void infixpre(char infix[])

{ int i=0,j=0,k,l;

char token,ch,prefix[100],p[100];

int isp(char);

int icp(char);

void presol(char[]);

push('\0');

for(l=0;infix[l]!='\0';l++);

--l;

token=infix[l];

for(i=l;i>=0;)

{ if(isalnum(token))

{

p[j]=token;

j++;

}

else if(token==')')

push(')');

else if(token=='(')

{

while(stack[top]!=')')

{

ch=pop();

p[j]=ch;

j++;

}

ch=pop();

}

else

{ if(!isalnum(infix[i]) && !isalnum(infix[i-1]))// && infix[i+1]!='(' && infix[i+1]!=')')

{ cout<<("\ninvalid infix");

exit(1);

}

while(isp(stack[top])>icp(token))

{

ch=pop();

p[j]=ch;

j++;

}

push(token);

}

i--;

token=infix[i];

}

while(top>=0)

{

ch=pop();

p[j]=ch;

j++;

}

--j;

for(i=0,l=--j;l>=0;l--,i++)

prefix[i]=p[l];

prefix[i]='\0';

cout<<"\n Infix to Pre Fix :";

for(k=0;prefix[k]!='\0';k++)

cout<<prefix[k];

//psol=postsol(postfix);

//cout<<"\nPostfix Notation solution is: "<<psol;

presol(prefix);

}

int isp(char token)

{ switch(token)

{

case ')':return(0);

case '(':return(9);

case '^':return(8);

case '\*':return(7);

case '/':return(7);

case '+':return(6);

case '-':return(6);

case '\0':return(0);

default:cout<< "\nillegaloperator"); exit(1);

}

return 0;

}

int icp(char token)

{ switch(token)

{

case '(':return(10);

case ')':return(9);

case '^':return(8);

case '\*':return(7);

case '/':return(7);

case '+':return(6);

case '-':return(6);

case '\0':return(0);

default: cout<<"\nillegaloperator"; exit(1);

}

return 0;

}

void presol(char prefix[])

{ int i,l,k,psol;

int ope(char);

for(l=0;prefix[l]!='\0';l++);

--l;

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

{

if(isdigit(prefix[i]))

push(prefix[i]-'0');

else

{ k=ope(prefix[i]); push(k); }

}

psol=pop();

cout<<"\nResult of prefix:"<<psol;

}

int ope(char token)

{ int a,b;

a=pop();b=pop();

switch(token)

{

case '\*':return a\*b;

case '/':return a/b;

case '^':return pow(a,b);

case '+':return a+b;

case '-':return a-b;

default: cout<<"\n illegal operator";

}

return 0;

}

# Example Program Of Infix To Postfix

#include<stdio.h>

#include<math.h>

#define max 100

int stack[max];

int top=-1;

void push(int a)

{

if(top==99)

cout<<"\nStack is Full";

else

{ top++;

stack[top]=a;

}

}

int pop()

{ int x;

if(top==-1)

cout<<"\nstack is full";

else

{ x=stack[top];

top--;

return x;

}

return 0;

}

void main()

{

char infix[100];

int postsol(char []);

void infixpost(char[]);

clrscr();

cout<<"enter the infixexpession :";

gets(infix);

infixpost(infix);

}

void infixpost(char infix[])

{ int i=0,j=0,k,psol;char token,ch,postfix[100];

int isp(char);

int icp(char);

int postsol(char[]);

push('\0');

token=infix[i];

while(token!='\0')

{ if(isalnum(token))

{

postfix[j]=token;

j++;

}

else if(token=='(')

push('(');

else if(token==')')

{

while(stack[top]!='(')

{

ch=pop();

postfix[j]=ch;

j++;

}

ch=pop();

}

else

{ if(icp(infix[i+1])<8)

{

cout<<"\noperators is more than operands";

exit(1);

}

while(isp(stack[top])>=icp(token))

{

ch=pop();

postfix[j]=ch;

j++;

}

push(token); }

i++;

token=infix[i];

}

while(top!=-1)

{

ch=pop();

postfix[j]=ch;

j++;

}

cout<<"\n Post Fix notation of expession:";

for(k=0;k<j;k++)

cout<<postfix[k];

psol=postsol(postfix);

cout<<"\nPostfix Notation solution is: "<<psol;

}

int isp(char token)

{

int i=12;

switch(token)

{

case '(':i=0;break;

case ')':i=9;break;

case '^':i=8;break;

case '\*':i=7;break;

case '/':i=7;break;

case '+':i=6;break;

case '-':i=6;break;

case '\0':i=0;break;

// default:

}

if(!isalnum(token) &&i==12 )

{ cout<<"\nillegal operator in isp";exit(1);}

return i;

}

int icp(char token)

{

int i=12;

switch(token)

{

case '(':i=10;break;

case ')':i=9;break;

case '^':i=8;break;

case '\*':i=7;break;

case '/':i=7;break;

case '+':i=6;break;

case '-':i=6;break;

case '\0':i=0;break;

// default:

}

if(!isalnum(token) &&i==12 )

{ cout<<"\nillegal operatorin icp" ;exit(1);}

return i;

}

int postsol(char postfix[])

{ int i,token,k;

int ope(char);

for(i=0;postfix[i]!='\0';i++)

{

if(!isalnum(postfix[i]))

{ token=postfix[i];

k=ope(token);

push(k);

}

else

{

token=postfix[i]-'0';

push(token);

}

}

}

int ope(char token)

{ int a,b;

b=pop();a=pop();

switch(token)

{

case '\*':return a\*b;

case '/':return a/b;

case '^':return pow(b,a);

case '+':return a+b;

case '-':return a-b;

}

}

**// program of simple Queue**

# include < iostream.h|>

# define MAX 3

int queue [ MAX];

int front = 0 , Rear = 0;

Void insert ( )

{int a;

Cout<" enter any element ";

Cin >> a ;

if (Rear = = MAX) cout<<" \ n queue is full";

else

{+ + Rear;

queue [Rear] = a ;

}

if (font = = 0 ) front + + ;

}

Void del ( )

{int x ;

if ( front = = 0 ) <<" \ n queue is empty " ;

else

{cout <<" deleted element is:";

x = queue [front];

Cout << x ;

if ( front = = Rear )

{front = 0 ; Rear = 0}

else

front + + ;

}

}

Main ( )

{int i, choice ;

for (i=1 ; i< = 100 ; + +i )

{Cout<<" \ n\ t\ t \*\*\*\* main menu \*\*\*\* \n ";

Cout <<" \ t [1] insert \n ";

Cout <<" \ t [2] delete \n ";

Cout <<" \ t [3] exit \n ";

Cout <<" \ n enter your choice:";

Cin>> choice

Switch (choice)

{Case 1: insert ( ); break;

Case 2: del ( ); break;

Case 3: exit ( );

default: Cout <<" \n invalid choice " ;

}

}

}

**// circular Queue**

# include < iostrem.h>

# define MAX 3 ;

int cque [MAX ] ;

int front = 0 , Rear = 0 ;

Void insert ( )

{int a ;

cout <<" enter any element ";

cin >> a;

if (( Rear = = MAX && front = = 1) //( front = Rear + 1))

cout<<" \ n cque is fall ";

else

if ( Rear = = MAX ) Rear = 1 ;

else

+ + Rear;

cque [ Rear ] = a ;

if ( front = = 0 ) front = 1 ;

}

void del ( )

{int x ;

if (front = = 0) <<" \ n cque is empty";

else

{cout <<" deleted element is: ";

x = cque [ front ]

cout << x ;

if ( front = = Rear )

{front = 0 ; Rear = 0 ;}

else

if ( front = MAX ) front = 1 ;

else

front + + ;

}

main ( )

{ int choice ;

While (1)

{ cout <<" \n \ t \ t \*\*\* main menu \*\*\* \ n " ;

cout <<" \ t [1] insert \ n " ;

cout <<" \t [2] delete \ n ";

cout <<" [3] exite \ n " ;

cout <<" \ n enter your choice : " ;

cin >> choice ;

Switch (choice)

{ case 1 : insert ( ) ; break ;

case 2 : del ( ) ; break ;

case 3 : break ;

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

}

}

}

**Program of create list and display**

# include < iostream.h>

# include < stdlib.h>

Typedef struck node

{

Int data ;

Struck node \* next ;

} node;

Node \* head=null;

Void create list ( )

{

Int ch ;

Node \* prev , \* temp ;

While ( 1)

{

Cout<<" \n enter your choice o – exit \ 1- add: ";

Cin >> ch ;

If ( ch = = 0 ) break ;

Temp = ( node \* ) malloc ( size of ( node ));

Cout <<" \n enter data for new node : ";

Cin >> temp data;

Temp next = null ;

If ( head = = null )

{ head = temp;

}

Else

{ prev next = temp ;

} prev = temp;

}

}

Void display ( )

{ node \* temp ;

For ( temp = head ; temp ! = null ; temp =temp next )

{ cout <<" \n data of list are at the address"

Cout <<temp data ;

}

Void main ( )

{ int ch ;

Hand :

Cout <<" \n [ 1 ] create list ";

Cout <<" \n [2 ] display list ";

Cout <<" \n [3 ] exit ";

Cout <<" \n enter your choice ";

Cin >> ch ;

Switch ( ch )

{ cas 1 : create list ( ) ; goto hand ;

Cas2: display ( ) ; goto hand ;

Cas3 : break;

}

}

## A Program of Binary Tree

#include<stdio.h>

#include<conio.h>

typedef struct node

{

int data;

struct node \*left, \*right;

}node;

node \*root=NULL;

node\* CreateTree(node \*ptr, int i, int k)

{

int ch;

if(ch==0)

cout<<“Wants to Create Root Node : “;

else if(i==1)

cout<<“Wants to Create Left Node of : “<< k;

else if(i==2)

cout<<“Wants to Create Right Node of %d : “<< k;

cin>>ch;

if(ch==0) break;

ptr=(node\*)malloc(sizeof(node));

cout<<“\nEnter Data :”;

cin>>prt->data;

ptr->left=CreateTree(NULL,1,ptr->data);

ptr->left=CreateTree(NULL,2,ptr->data);

return ptr;

}

void Inorder(node \*ptr)

{

if(ptr!=NULL)

{

inorder(ptr->left);

cout<<“\n” <<ptr->data;

inorder(ptr->right);

}

}

void preorder(node \*ptr)

{

if(ptr!=NULL)

{

cout<<“\n”<< ptr->data;

preorder(ptr->left);

preorder(ptr->right);

}

}

void postorder(node \*ptr)

{

if(ptr!=NULL)

{

postorder(ptr->left);

postorder(ptr->right);

cout<<“\n” <<ptr->data;

}

}

void main()

{

clrscr();

root=CreateTree(NULL,0,0);

cout<<“\nInorder : \n”;

inorder(root);

cout<<“\nPreorder : \n”;

preorder(root);

cout<<“\nPostorder : \n”;

postorder(root);

getch();

}

# Binary Search Tree

#include<stdio.h>

#include<conio.h>

struct tree

{

int data;

struct tree \*left;

struct tree \*right;

};

typedef struct tree node;

node \*root=NULL;

node\* createnode()

{ node \*temp;

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

temp->left=NULL;

temp->right=NULL;

return temp;

}

node \*createtree(node \*root)

{ node \*temp,\*temp1;

char ch='y';

root=createnode();

cout<<“Enter data for root:”;

cin>>root->data;

while(ch=='y')

{ temp=root;

temp1=createnode();

flushall();

cout<<“\nEnter data for item:”;

cin>>temp1->data;

while(1)

{ if(temp1->data==temp->data)

{ cout<<“\Duplicate is not allow here”;

break;

}

if(temp1->data<temp->data)

{ if(temp->left==NULL)

{ temp->left=temp1;

break;

}

else

temp=temp->left;

}

else

{ if(temp->right==NULL)

{ temp->right=temp1;

break;

}

else

temp=temp->right;

}

}

flushall();

cout<<“\tWants to enter more [y/n]:”;

ch=getche();

}

return root;

}

void insert(int key)

{ node \*temp,\*temp1;

temp=root;

temp1=createnode();

flushall();

temp1->data=key;

cout<<“\nEnter data for item:”;

cin>>temp1->data;

while(1)

{ if(temp1->data==temp->data)

{ cout<<“\Duplicate is not allow here”;

break;

}

if(temp1->data<temp->data)

{ if(temp->left==NULL)

{ temp->left=temp1;

break;

}

else

temp=temp->left;

}

else

{ if(temp->right==NULL)

{ temp->right=temp1;

break;

}

else

temp=temp->right;

}

}//end of while

}//end of insert

void deletion(node \*t,int item)

{ node \*q=NULL,\*rp,\*f,\*s;

while(t!=NULL && t->data!=item)

{ q=t;

t=(item<t->data) ? t->left :t->right;

}

if(t==NULL)

return;

if(t->left==NULL)

rp=t->right;

else

if(t->right==NULL)

rp=t->left;

else

{ f=t;

rp=t->right;

s=rp->left;

while(s!=NULL)

{ f=rp;

rp=s;

s=rp->left;

}

if(f!=t)

{ f->left=rp->right;

rp->right=t->right;

}

rp->left=t->left;

}

if(q==NULL)

t=rp;

else

if(t==q->left )

q->left=rp;

else

q->right=rp;

return;

}

void inorder(node \*cnode)

{ if(cnode!=NULL)

{ inorder(cnode->left);

// cout<<“\n”;

cout<<“\t”<< cnode->data;

inorder(cnode->right);

}

}

void main()

{ int key,item,ch;

clrscr();

while (1)

{ clrscr();

cout<<“\n\t#############BINARYSEARCH TREE##############\n\n”;

cout<<“\t [1] CREATE BIONARY SEARCH TREE\n”;

cout<<“\t [2] INSERT\n”;

cout<<“\t [3] INORDER TRAVERSER\n”;

cout<<“\t [4] DELETION\n”;

cout<<“\t [5] EXIT\n\n”;

cout<<“Enter your choice [1-8] “;

cin>>ch;

switch(ch)

{

case 1:{ root=createtree(root);

break;

}

case 2:{ cout<<“\n\nEnter item to be inserted :”;

cin>>key;

insert(key); break;

}

case 3:inorder(root); getch(); break;

case 4:{ cout<<“\n\nEnter item to be deleted :”;

cin>>item;

deletion(root,item);

break;

}

case 5:exit(1);

}//end of switch

}//end of while

getch();

}