|  |  |
| --- | --- |
| C:\Users\ITLAB\Downloads\vjit11.png | **Vidya Jyothi Institute of Technology**  (Approved by AICTE, New Delhi, Accredited by NAAC, Permanently Affiliated to JNTUH, Hyderabad)  **An Autonomous Institution**  Aziznagar Gate, Chilkur Balaji Road,  Hyderabad – 500075, Telangana, India  **www.vjit.ac.in** |

**Department of Information Technology**

# Course Name : Data Structures Lab

Course ID : A13585

Prerequisites : C Programming

# 

II B Tech – I Semester

(R15)

MR. B ESWAR BABU

Associate Professor

Course Coordinator

**LABORATORY COURSE FILE INDEX**

|  |  |
| --- | --- |
| **S. No.** | **Description** |
| 1 | Program Outcomes & Laboratory Course Outcomes |
| 2 | Mapping of course outcomes (CO’S) with program outcomes (PO’S) & program specific outcomes (PSO’S) |
| 3 | Time Table |
| 4 | List of Experiments/Learning outcomes of each experiment |
| 5 | List of Equipment in Laboratory |
| 6 | Lab Experiment Schedule |
| 7 | Laboratory Manual |
| 8 | Rubrics for evaluation of Experimental work |
| 9 | Direct Attainment |
| 10 | Course Exit Survey for Indirect Attainment |
| 11 | Course Closure Report |

**Program Outcomes (POs)**

1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**Program Specific Outcomes (PSOs)**

1. Use mathematical abstractions and algorithmic design along with programming tools to solve complexities involved in efficient programming.
2. To provide software skills and documentation ability for graduates to become employable/ higher studies/ Entrepreneur/Researcher.

**Course Outcomes (Cos)**

|  |  |
| --- | --- |
| **After completing this course the student must demonstrate the knowledge and ability to** | |
| **CO1** | Develop the programs on stack and its applications |
| **CO2** | Demonstrate the operations on trees |
| **CO3** | Demonstrate the implementations of various advanced trees |
| **CO4** | Design and implementation of programs on BST and graph traversals |
| **CO5** | Understand the C++ program structure and also basics of C++ programming. |

**CO –PO Mapping:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PO 1** | **PO 2** | **PO 3** | **PO 4** | **PO 5** | **PO 6** | **PO 7** | **PO 8** | **PO 9** | **PO 10** | **PO 11** | **PO 12** |
| **CO 1** | 3 | 3 | 3 | 3 | 3 | 2 |  |  | 2 | 2 | 1 | 3 |
| **CO 2** | 3 | 3 | 3 | 3 | 3 | 2 |  |  | 2 | 2 | 1 | 3 |
| **CO 3** | 3 | 3 | 3 | 3 | 3 | 2 |  |  | 2 | 2 | 1 | 3 |
| **CO 4** | 3 | 3 | 3 | 3 | 3 | 2 |  |  | 2 | 2 | 1 | 3 |
| **CO 5** | 3 | 3 | 3 | 3 | 3 | 2 |  |  | 2 | 2 | 1 | 3 |

**CO - PSO Mapping:**

|  |  |  |
| --- | --- | --- |
|  | **PSO1** | **PSO2** |
| **CO1** | 3 | 3 |
| **CO2** | 3 | 3 |
| **CO3** | 3 | 3 |
| **CO4** | 3 | 3 |
| **CO5** | 3 | 3 |

**Time Table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **9:30AM -10:30AM** | **10:30AM-**  **11:30AM** | **11:30 AM-12:30PM** | | **12:30-1:15PM** | **1:15PM-2:15PM** | **2:15PM-3:15 PM** | **3:15PM-4:15PM** |
| **MON** |  |  |  | | **LUNCH** |  |  |  |
| **TUE** |  |  |  | |  | **DS Lab** | |
| **WED** |  |  |  | |  |  |  |
| **THU** |  |  |  | |  |  |  |
| **FRI** |  |  | |  |  |  |  |
| **SAT** |  | | | |  | | |

**List of Experiments / Learning outcomes of each experiment**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Name of the Experiment** | **Learning Outcomes** |
|  | **Part-A** |  |
| 1 | Write a program to illustrate string built in functions | Use appropriate string built in functions to manipulate the strings. |
| 2 | Write a program to evaluate postfix notations | Apply Stack algorithm strategy and design algorithm to evaluate Postfix expressions. |
| 3 | Write a program to convert infix to post fix notation | Apply Stack algorithm strategy and design algorithm to convert infix expression to postfix expressions. |
| 4 | Write a program to illustrate tree traversals  a)In order b)Preorder c)Post order | Apply tree traversal methods in real time applications. |
| 5 | Write a program to illustrate insertion, deletion and searching in Binary Search Tree. | Design source code to implement features of a Binary Search Tree, such as insertion, deletion and Searching |
| 6 | Write a program to illustrate Graph traversals   1. Breadth First Search 2. Depth First Search | Apply Graph traversal algorithms in real time applications. |
| 7 | Write a program to illustrate Insertion, deletion and Rotation on AVL Trees. | Learn the computational limitations of BST and solve the limitations using AVL trees in balancing the height of a tree. |
|  | **Part - B** |  |
| 1 | Program to illustrate Function overloading to calculate area of circle, rectangle and square. | To learn how to overload functions in C++. |
| 2 | Program to illustrate Virtual function | To learn how virtual functions implement dynamic binding with polymorphism. |
| 3 | Program to illustrate default constructor, parameterized constructor, copy constructors. | To learn how to implement constructors and class member functions. |
| 4 | Program to illustrate single inheritance, multiple inheritance, multilevel inheritance, Hybrid inheritance. | To illustrate the feature of inheritance in C++. |
| 5 | Program to illustrate run time polymorphism, compile time polymorphism. | To learn how inheritance and virtual functions implement dynamic binding with polymorphism. |
| 6 | Program to illustrate Operator overloading   1. Unary operator b) Binary operator | To learn how to overload operators in C++. |
| 7 | Program to illustrate Exception handling mechanisms using try, catch, throw key words. | To learn how to use exception handling in C++ programs. |
| 8 | Program to illustrate formatted and unformatted I/O streams. | To learn how to use advanced features of C++ specifically stream I/O |

**List of Equipment in Laboratory**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Description of the Equipment** | **Quantity** |
| 1 | Acer Veriton Desktop I3 4170  4GB DDR3 RAM  500 GB HDD  LED Monitor | 60 |
| 2 | Dev C++ | 60 |

.

**Lab Experiment Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Name of the Experiment** | **Scheduled Date** | **Completed Date** |
|  | **Part-A** |  |  |
| 1 | Write a program to illustrate string built in functions |  |  |
| 2 | Write a program to evaluate postfix notations |  |  |
| 3 | Write a program to convert infix to post fix notation |  |  |
| 4 | Write a program to illustrate tree traversals  a)In order b)Preorder c)Post order |  |  |
| 5 | Write a program to illustrate insertion, deletion and searching in Binary Search Tree. |  |  |
| 6 | Write a program to illustrate Graph traversals   1. Breadth First Search 2. Depth First Search |  |  |
| 7 | Write a program to illustrate Insertion, deletion and Rotation on AVL Trees. |  |  |
|  | **Part - B** |  |  |
| **1** | Program to illustrate Function overloading to calculate area of circle, rectangle and square. |  |  |
| **2** | Program to illustrate Virtual function |  |  |
| **3** | Program to illustrate default constructor, parameterized constructor, copy constructors. |  |  |
| **4** | Program to illustrate single inheritance, multiple inheritance, multilevel inheritance, Hybrid inheritance. |  |  |
| **5** | Program to illustrate run time polymorphism, compile time polymorphism. |  |  |
| **6** | Program to illustrate Operator overloading   1. Unary operator b) Binary operator |  |  |
| **7** | Program to illustrate Exception handling mechanisms using try, catch, throw key words. |  |  |
| **8** | Program to illustrate formatted and unformatted I/O streams. |  |  |

**Experiment - 1**

**Aim: Program illustrate String handling functions**

**Strlen():**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char a[20];

int n;

printf(“enter the string: \n”);

gets(a);

n=strlen(a);

printf("Length of string b = %d \n", n);

getch();

}

**OUTPUT:**

Enter the string: devakishan

Length of string a = 10

**Strcpy():**

#include <stdio.h>

#include <string.h>

void main()

{

char str1[10]= "awesome";

char str2[10];

char str3[10];

strcpy(str2, str1);

strcpy(str3, "well");

puts(str2);

puts(str3);

getch();

}

**OUTPUT:**

awesome

well

**strcat():**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char a[20],b[20];

printf(“enter the first string: \n”);

gets(a);

printf(“enter the second string: \n”);

gets(b);

strcat(a,b);

printf(“the after concatenating is: \n”,a);

getch();

}

**OUTPUT:**

Enter the first string: Eswar

Enter the second string: Banala

the after concatenating is: EswarBanala

**Experiment 2**

**Aim: Write a C program to evaluate postfix notations**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#define MAX 50

int stack[MAX];

char postfix[20];

int top=-1;

void push(int item);

char pop();

void evaluate(char s);

void main()

{

int i,n,x;

char symbol;

// clrscr();

printf("Insert a postfix notation :: \n");

gets(postfix);

n=strlen(postfix);

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

{

symbol=postfix[i];

if(symbol>='0' && symbol<='9')

{

x=(int)(symbol-48);

push(x);

}

else

{

evaluate(postfix[i]);

}

}

printf("\n\nResult is :: %d",stack[top]);

getch();

}

void push(int x)

{

top++;

stack[top]=x;

}

char pop()

{

char x;

x=stack[top];

top--;

return x;

}

void evaluate(char d)

{

int a,b,c;

a=pop();

b=pop();

switch(d)

{

case '+':

c=a+b;

top++;

stack[top]=c;

break;

case '-':

c=a-b;

top++;

stack[top]=c;

break;

case '\*':

c=a\*b;

top++;

stack[top]=c;

break;

case '/':

c=a/b;

top++;

stack[top]=c;

break;

case '%':

c=a%b;

break;

top++;

stack[top]=c;

default:

c=0;

}

}

**OUTPUT:**

Insert a postfix notation: 432+\*

Result is: 20

**Experiment 3**

**Aim: Write a C program that uses stack operations to convert a given infix expression into its postfix Equivalent, Implement the stack using an array.**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<conio.h>

char stack[20];

int top=-1;

char pop(); /\*declaration of pop function\*/

void push(char item); /\*declaration of push function\*/

int prcd(char symbol) /\*checking the precedence\*/

{

switch(symbol) /\*assigning values for symbols\*/

{

case '+':

case '-': return 2;

break;

case '\*':

case '/':

case '%': return 4;

break;

case '(':

case ')':

case '#': return 1;

break;

}

}

int isoperator(char symbol) /\*assigning operators\*/

{

switch(symbol)

{

case '+':

case '\*':

case '-':

case '/':

case '%':

case '(':

case ')':return 1;

break;

default: return 0;

}

}

/\*converting infix to postfix\*/

void convertip(char infix[],char postfix[])

{

int i,n,j=0;

char symbol;

top++;

stack[top]='#';

n=strlen(infix);

for(i=0;i<strlen(infix);i++)

{

symbol=infix[i];

if(isoperator(symbol)==0)

{

postfix[j]=symbol;

j++;

}

else

{

if(symbol=='(')

push(symbol); /\*function call for pushing elements into the stack\*/

else if(symbol==')')

{

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

{

postfix[j]=pop();

j++;

}

pop(); /\*function call for popping elements into the stack\*/

}

else

{

if(prcd(symbol)>prcd(stack[top]))

push(symbol);

else

{

while(prcd(symbol)<=prcd(stack[top]))

{

postfix[j]=pop();

j++;

}

push(symbol);

}/\*end of else \*/

}/\*end of else \*/

} /\*end of else \*/

}/\*end of for loop\*/

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

{

postfix[j]=pop();

j++;

}

postfix[j]='\0'; /\*null terminate string\*/

}

/\*main program\*/

void main()

{

char infix[20],postfix[20];

//clrscr();

printf("enter the valid infix string \n");

gets(infix);

convertip(infix,postfix); /\*function call for converting infix to postfix \*/

printf("the corresponding postfix string is:\n");

puts(postfix);

getch();

}

/\*push operation\*/

void push(char item)

{

top++;

stack[top]=item;

}

/\*pop operation\*/

char pop()

{

char a;

a=stack[top];

top--;

return a;

}

**Output:**

enter the valid infix string

9-((3\*4)+8/4

the corresponding postfix string is: 934\*8+-4/

**Experiment 4**

**Aim: C program to illustrate tree traversals**

1. **In order b) preorder c) post order**

In-order traversal method:  
1. Visit Left Sub-Tree  
2. Process Current Node  
3. Visit Right Sub-Tree  
  
Pre-order traversal method:  
1. Process Current Node  
2. Visit Left Sub-Tree  
3. Visit Right Sub-Tree  
  
Post-order traversal method:  
1. Visit Left Sub-Tree  
2. Visit Right Sub-Tree  
3. Process Current Node

#include <stdio.h>

#include <stdlib.h>

struct node

{

int value;

node\* left;

node\* right;

};

struct node\* root;

struct node\* insert(struct node\* r, int data);

void inOrder(struct node\* r);

void preOrder(struct node\* r);

void postOrder(struct node\* r);

int main()

{

root = NULL;

int n, v;

printf("How many data's do you want to insert ?\n");

scanf("%d", &n);

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

printf("Data %d: ", i+1);

scanf("%d", &v);

root = insert(root, v);

}

printf("Inorder Traversal: ");

inOrder(root);

printf("\n");

printf("Preorder Traversal: ");

preOrder(root);

printf("\n");

printf("Postorder Traversal: ");

postOrder(root);

printf("\n");

return 0;

}

struct node\* insert(struct node\* r, int data)

{

if(r==NULL)

{

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

r->value = data;

r->left = NULL;

r->right = NULL;

}

else if(data < r->value){

r->left = insert(r->left, data);

}

else {

r->right = insert(r->right, data);

}

return r;

}

void inOrder(struct node\* r)

{

if(r!=NULL){

inOrder(r->left);

printf("%d ", r->value);

inOrder(r->right);

}

}

void preOrder(struct node\* r)

{

if(r!=NULL){

printf("%d ", r->value);

preOrder(r->left);

preOrder(r->right);

}

}

void postOrder(struct node\* r)

{

if(r!=NULL){

postOrder(r->left);

postOrder(r->right);

printf("%d ", r->value);

}

}

**OUTPUT:**

How many data's do you want to insert? 7

20 15 25 16 24 12 30

Inorder Traversal: 12 15 16 20 24 25 30

Preorder Traversal: 20 15 12 16 25 24 30

Postorder Traversal: 12 16 15 24 30 25 20

**Experiment 5**

**Aim: C program to illustrate insert, delete and searching operation in binary search tree**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int key;

struct node \*left, \*right;

};

// A utility function to create a new BST node

struct node \*newNode(int item)

{

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

temp->key = item;

temp->left = NULL;

temp->right = NULL;

return temp;

}

// A utility function to do inorder traversal of BST

void inorder(struct node \*root)

{

if (root != NULL)

{

inorder(root->left);

printf("%d ", root->key);

inorder(root->right);

}

}

struct node\* insert(struct node\* root, int key)

{

/\* If the tree is empty, return a new node \*/

if (root == NULL)

return newNode(key);

if (key < root->key)

root->left = insert(root->left, key);

else

root->right = insert(root->right, key);

/\* return the (unchanged) node pointer \*/

return root;

}

struct node \* Findmin(struct node\* node)

{

struct node\* current = node;

/\* loop down to find the leftmost leaf \*/

while (current->left != NULL)

current = current->left;

return current;

}

struct node\* deletion(struct node\* root, int key)

{

// base case

if (root == NULL) return root;

// If the key to be deleted is smaller than the root's key,

// then it lies in left subtree

if (key < root->key)

root->left = deletion(root->left, key);

// If the key to be deleted is greater than the root's key,

// then it lies in right subtree

else if (key > root->key)

root->right = deletion(root->right, key);

else

{

// node with only one child or no child

if (root->left == NULL)

{

struct node \*temp = root->right;

free(root);

return temp;

}

else if (root->right == NULL)

{

struct node \*temp = root->left;

free(root);

return temp;

}

// smallest element from thee right subtree

struct node\* temp = Findmin(root->right);

// Copy the inorder successor's content to this node

root->key = temp->key;

// Delete the inorder successor

root->right = deletion(root->right, temp->key);

}

return root;

}

// main program

void main()

{

int data,ch;

struct node \*root = NULL;

while(1)

{

printf("\nenetr 1.insertion 2.display 3.deletion. 4 exit\n");

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nenter the value to be inserted\n");

scanf("%d",&data);

root = insert(root, data);

break;

case 2: printf("\nInorder traversal of the given tree is: \n");

inorder(root);

break;

case 3: printf("\nenter the value to be deleted\n");

scanf("%d",&data);

root = deletion(root, data);

break;

case 4: exit(0);

}

}

}

**OUTPUT:**

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

1

enter the value to be inserted

20

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

1

enter the value to be inserted

25

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

1

enter the value to be inserted

15

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

1

enter the value to be inserted

12

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

1

enter the value to be inserted

16

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

2

enter the value to be deleted

15

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

2

Inorder traversal of the given tree is: 12 16 20 24 25 30

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

4

**Experiment 6**

**Aim:Write a ‘C’ program to implement Breadth first search.**

#include <stdio.h>

#include<conio.h>

void creategraph();

void bfs(); void display();

int g[10][10],n;

void main()

{

int v;

clrscr();

creategraph();

printf(“starting vertex is”);

scanf(“%d”,&v);

bfs(v);

getch();

}

void creategraph()

{

int i,j;

printf(“enter the number of nodes”);

scanf(“%d”,&n);

for(i=0;i”,v);

while(f<=r)

{

v=q[f];

f++;

for(i=0;i”,i);

visited[i]=1;

q[++r]=i;

}

}

**Output:**

Enter no of nodes 3

Edge present between A&A 0

Edge present between A&B 1

Edge present between A&C 1

Edge present between B&A 1

Edge present between B&B 0

Edge present between B&C 1

Edge present between C&A 1

Edge present between C&B 1

Edge present between C&C 0

Enter vertex from abopve graph 0

**A B C**

**A 0 1 1**

**B 1 0 1**

**C 1 1 0**

**Aim: Write a ‘C’ program to implement Depth first search.**

#include<stdio.h>

#include <conio.h>

void creategraph();

void dfs();

void display();

int g[10][10],n;

void main()

{

int v;

clrscr();

creategraph();

printf(“starting vertex is”);

scanf(“%d”,&v);

dfs(v);

getch();

}

void creategraph()

{

int i,j;

printf(“enter the number of nodes”);

scanf(“%d”,&n);

for(i=0;i=0)

{

v=st[top];

top--;

if(visited [v]==0)

{

printf(“%d->”,v);

visited [v]=1;

}

for(i=n-1;i>=0;i--)

{

if(g[v][i]!=0 & & visited[i]==0)

{

st[++top]=i;

}

}}}

**Output:**

Enter no.of nodes 3

Edge present b/w A&A 0

Edge present b/w A&B 1

Edge present b/w A&C 1

Edge present b/w A&D 0

Edge present b/w B&A 1

Edge present b/w B&B 0

Edge present b/w B&C 0

Edge present b/w B&D 1

Edge present b/w C&A 1

Edge present b/w C&B 0

Edge present b/w C&C 0

Edge present b/w C&D 0

Edge present b/w D&A 0

Edge present b/w D&B 1

Edge present b/w D&C 0

Edge present b/w D&D 0

Enter vertex from above graph

0

A->B->D->C->

**A B C D**

**A 0 1 1 0**

**B 1 0 0 1**

**C 1 0 0 0**

**D 0 1 0 0**

**Experiment 7**

**Aim: C Program to illustrate insertion, deletion and rotations on AVL Trees**

#include<stdio.h>

#include<stdlib.h>

// An AVL tree node

struct node

{

int key;

struct node \*left;

struct node \*right;

int height;

};

// A utility function to get maximum of two integers

int max(int a, int b);

// A utility function to get height of the tree

int height(struct node \*N)

{

if (N == NULL)

return 0;

return N->height;

}

// A utility function to get maximum of two integers

int max(int a, int b)

{

return (a > b)? a : b;

}

/\* Helper function that allocates a new node with the given key and

NULL left and right pointers. \*/

struct node\* newNode(int key)

{

struct node\* node = (struct node\*)

malloc(sizeof(struct node));

node->key = key;

node->left = NULL;

node->right = NULL;

node->height = 1; // new node is initially added at leaf

return(node);

}

// A utility function to right rotate subtree rooted with y

// See the diagram given above.

struct node \*rightRotate(struct node \*y)

{

struct node \*x = y->left;

struct node \*T2 = x->right;

// Perform rotation

x->right = y;

y->left = T2;

// Update heights

y->height = max(height(y->left), height(y->right))+1;

x->height = max(height(x->left), height(x->right))+1;

// Return new root

return x;

}

// A utility function to left rotate subtree rooted with x

// See the diagram given above.

struct node \*leftRotate(struct node \*x)

{

struct node \*y = x->right;

struct node \*T2 = y->left;

// Perform rotation

y->left = x;

x->right = T2;

// Update heights

x->height = max(height(x->left), height(x->right))+1;

y->height = max(height(y->left), height(y->right))+1;

// Return new root

return y;

}

// Get Balance factor of node N

int getBalance(struct node \*N)

{

if (N == NULL)

return 0;

return height(N->left) - height(N->right);

}

struct node\* insert(struct node\* node, int key)

{

/\* 1. Perform the normal BST rotation \*/

if (node == NULL)

return(newNode(key));

if (key < node->key)

node->left = insert(node->left, key);

else

node->right = insert(node->right, key);

/\* 2. Update height of this ancestor node \*/

node->height = max(height(node->left), height(node->right)) + 1;

/\* 3. Get the balance factor of this ancestor node to check whether

this node became unbalanced \*/

int balance = getBalance(node);

// If this node becomes unbalanced, then there are 4 cases

// Left Left Case

if (balance > 1 && key < node->left->key)

return rightRotate(node);

// Right Right Case

if (balance < -1 && key > node->right->key)

return leftRotate(node);

// Left Right Case

if (balance > 1 && key > node->left->key)

{

node->left = leftRotate(node->left);

return rightRotate(node);

}

// Right Left Case

if (balance < -1 && key < node->right->key)

{

node->right = rightRotate(node->right);

return leftRotate(node);

}

/\* return the (unchanged) node pointer \*/

return node;

}

// A utility function to print preorder traversal of the tree.

// The function also prints height of every node

void preOrder(struct node \*root)

{

if(root != NULL)

{

printf("%d ", root->key);

preOrder(root->left);

preOrder(root->right);

}

}

/\* Drier program to test above function\*/

void main()

{

int data,ch;

struct node \*root = NULL;

while(1)

{

printf("\nenetr 1.insertion 2.display 3.deletion. 4 exit\n");

printf("enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nenter the value to be inserted\n");

scanf("%d",&data);

root = insert(root, data);

break;

case 2: printf("\n preorder traversal of the given tree is: \n");

preOrder(root);

break;

/\*case 3: printf("\nenter the value to be deleted\n");

scanf("%d",&data);

root = deletion(root, data);

break;\*/

case 4: exit(0);

}

}

}

**OUTPUT:**

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

1

20

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

1

15

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

1

12

enetr 1.insertion 2.display 3.deletion. 4 exit

enter your choice

2

preorder traversal of the given tree is:

15 12 20

enetr 1.insertion 2.display 3.deletion. 4 exit

4

**Part – B**

**Experiment 1**

**Program to illustrate function overloading to calculate area of circle,rectangle and square**.

#include<iostream>

using namespace std;

class sample

{

public:int res;

void area(int a)

{

res=a\*a;

cout<<"area of square is :";

cout<<res;

}

void area(int a,int b)

{

res=a\*b;

cout<<"\n area of rectangle is :";

cout<<res;

}

void area(float a,int b)

{

res=a\*b\*b;

cout<<"\n area of circle is: ";

cout<<res;

}

};

int main()

{

sample s;

int n,m;

cout<<"enter side of square\n";

cin>>n;

s.area(n);

cout<<"\n enter length and breadth of rectangle\n";

cin>>n>>m;

s.area(n,m);

cout<<"\n enter radius of circle\n";

cin>>n;

float pie=3.14;

s.area(pie,n);

}

**output:**

enter side of square

2

area of square is :4

enter length and breadth of rectangle

2

5

area of rectangle is :10

enter radius of circle

2

area of circle is: 12

**Experiment 2**

**Program to illustrate virtual function.**

#include<iostream>

using namespace std;

class base

{

public:

virtual void show()

{

cout<<"base class";

}

};

class derived:public base

{

public:

void show()

{

cout<<"derived class";

}

};

int main()

{

base \*b;

derived d;

b=&d;

b->show();

}

**output:** derived class

**Experiment 3**

**Program to illustrate**

**a)default constructor.**

#include<iostream>

using namespace std;

class test

{

int a,b;

public:

test()

{

cout<<"\n object has been created \n";

a=0;

b=0;

}

void display()

{

cout<<"\n a : "<<a;

cout<<"\n b: "<<b;

}

};

int main()

{

test t;

t.display();

}

**output:**

object has been created

a : 0

b: 0

**b) parameterised constructor.**

#include<iostream>

using namespace std;

class test

{

int a,b;

public:

test();

test(int ,int);

void display();

};

test::test()

{

cout<<"\n object has been created \n";

a=0;

b=0;

}

test::test(int x,int y)

{

a=x;

b=y;

}

void test::display()

{

cout<<"\n a: "<<a;

cout<<"\n b: "<<b;

}

int main()

{

test t(10,20);

t.display();

}

**output:**

a: 10

b: 20

**c)copy constructor.**

#include<iostream>

using namespace std;

class test

{

int a,b;

public:

test(int,int);

void display();

};

test::test(int x,int y)

{

cout<<"\n object has been created\n";

a=x;

b=y;

}

void test::display()

{

cout<<"\n a: "<<a;

cout<<"\n b: "<<b;

}

int main()

{

test t1(10,20);

test t2=t1;

t1.display();

t2.display();

}

**output:**

object has been created

a: 10

b: 20

a: 10

b: 20

**Experiment 4**

**Program to illustrate**

**a)single inheritance**

#include<iostream>

using namespace std;

class A

{

private:

int a;

public:

int b;

void getab();

int geta();

void showa();

};

class B:public A

{

int c;

public:

void mul();

void display();

};

void A::getab()

{

cout<<"\n read a,b ";

cin>>a>>b;

}

int A::geta()

{

return a;

}

void A::showa()

{

cout<<"\n a= "<<a;

}

void B::mul()

{

c=b\*geta();

}

void B::display()

{

cout<<"\n a= "<<geta();

cout<<"\n b= "<<b;

cout<<"\n c= "<<c;

}

int main()

{

B b;

b.getab();

b.mul();

b.showa();

b.display();

b.b=20;

b.mul();

b.display();

}

**output:**

read a,b

3

4

a= 3

a= 3

b= 4

c= 12

a= 3

b= 20

c= 60

**b)multiple inheritance**.

#include<iostream>

using namespace std;

class A

{

protected:

int rollno;

char name[20];

};

class B

{

protected:

int s1,s2,s3;

};

class compute:public A,public B

{

private:

float total,avg;

public:

void read();

void computetot();

void display();

};

void compute::read()

{

cout<<"\n read rollno: ";

cin>>rollno;

cout<<"\n read name: ";

cin>>name;

cout<<"\n read s1,s2,s3 ";

cin>>s1>>s2>>s3;

}

void compute::display()

{

cout<<"\n name : "<<name;

cout<<"\n no : "<<rollno;

cout<<"\n s1 : "<<s1;

cout<<"\n s2 : "<<s2;

cout<<"\n s3 : "<<s3;

cout<<"\n total: "<<total;

cout<<"\n avg : "<<avg;

}

void compute::computetot()

{

total=s1+s2+s3;

avg=total/3;

}

int main()

{

compute p;

p.read();

p.computetot();

p.display();

}

**output:**

read rollno: 12

read name: abc

read s1,s2,s3

99

99

99

name : abc

no : 12

s1 : 99

s2 : 99

s3 : 99

total: 297

avg : 99

**c)multi-level inheritance.**

#include<iostream>

using namespace std;

class student

{

protected:

int rollno;

char name[20];

public:

void read();

};

class marks:public student

{

protected:

int s1,s2,s3;

public:

void read();

};

class compute:public marks

{

private:

float total,avg;

public:

void computetotal();

void display();

};

void student::read()

{

cout<<"\n read rollno: ";

cin>>rollno;

cout<<"\n read name: ";

cin>>name;

}

void marks::read()

{

cout<<"\n read s1,s2,s3 ";

cin>>s1>>s2>>s3;

}

void compute::computetotal()

{

total=s1+s2+s3;

avg=total/3;

}

void compute::display()

{

cout<<"\n name : "<<name;

cout<<"\n no : "<<rollno;

cout<<"\n s1 : "<<s1;

cout<<"\n s2 : "<<s2;

cout<<"\n s3 : "<<s3;

cout<<"\n total: "<<total;

cout<<"\n avg : "<<avg;

}

int main()

{

compute c;

c.student::read();

c.marks::read();

c.computetotal();

c.display();

}

**output:**

read rollno: 12

read name: abc

read s1,s2,s3

99

99

99

name : abc

no : 12

s1 : 99

s2 : 99

s3 : 99

total: 297

avg : 99

**d)hybrid inheritance**.

#include<iostream>

using namespace std;

class student

{

protected:

int rollno;

char name[20],branch[10];

};

class inmarks:virtual public student

{

protected:

int s1,s2,s3;

};

class exmarks:virtual public student

{

protected:

int s1,s2,s3;

};

class result:public inmarks,public exmarks

{

private:

int s1,s2,s3;

float tot;

public:

void readdata()

{

cout<<"\n read no ";

cin>>rollno;

cout<<"\n read name ";

cin>>name;

cout<<"\n read branch ";

cin>>branch;

cout<<"\n read inmarks s1 ";

cin>>inmarks::s1;

cout<<"\n read inmarks s2 ";

cin>>inmarks::s2;

cout<<"\n read inmarks s3 ";

cin>>inmarks::s3;

cout<<"\n read exmarks s1 ";

cin>>exmarks::s1;

cout<<"\n read exmarks s2 ";

cin>>exmarks::s2;

cout<<"\n read exmarks s3 ";

cin>>exmarks::s3;

}

void display()

{

cout<<"\n name: "<<name;

cout<<"\n rollno: "<<rollno;

cout<<"\n branch: "<<branch;

cout<<"\n inmarks s1: "<<inmarks::s1;

cout<<"\n inmarks s2: "<<inmarks::s2;

cout<<"\n inmarks s3: "<<inmarks::s3;

cout<<"\n exmarks s1: "<<exmarks::s1;

cout<<"\n exmarks s2: "<<exmarks::s2;

cout<<"\n exmarks s3: "<<exmarks::s3;

cout<<"\n total: "<<tot;

}

void compute()

{

s1=inmarks::s1+exmarks::s1;

s2=inmarks::s2+exmarks::s2;

s3=inmarks::s3+exmarks::s3;

tot=s1+s2+s3;

cout<<tot;

}

};

int main()

{

result r;

r.readdata();

r.compute();

r.display();

}

**output:**

read no 12

read name abc

read branch cse

read inmarks s1 25

read inmarks s2 25

read inmarks s3 25

read exmarks s1 75

read exmarks s2 75

read exmarks s3 75

300

name: abc

rollno: 12

branch: cse

inmarks s1: 25

inmarks s2: 25

inmarks s3: 25

exmarks s1: 75

exmarks s2: 75

exmarks s3: 75

total: 300

**Experiment 5**

**Program to illustrate runtime, compiletime polymorphism.**

1. **compile time polymorphism:**

#include<iostream>

using namespace std;

class addition

{

public:

void sum(int x)

{

int y;

cin>>y;

cout<<"\naddition of 1st function= \n"<<x+y;

}

void sum(int x,int y)

{

cout<<"\naddition of 2nd function= \n"<<x+y;

}

void sum(int x,int y,int z)

{

cout<<"\naddition of 3rd function= \n"<<x+y+z;

}

};

int main()

{

addition a;

a.sum(10);

a.sum(10,20);

a.sum(10,20,30);

}

**output:**

5

addition of 1st function=

15

addition of 2nd function=

30

addition of 3rd function= 60

**b)runtime polymorphism:**

#include<iostream>

using namespace std;

class base

{

public:

void show()

{

cout<<"base class";

}

};

class derived:public base

{

public:

void show()

{

cout<<"derived class";

}

};

int main()

{

base \*b;

derived d;

b=&d;

b->show();

}

**output:**

base class

**Experiment 6**

**Program to illustrate operator overloading**

**a)unary operator.**

#include<iostream>

using namespace std;

class A

{

int x,y,z;

public:

A(int n,int m,int k)

{

x=n;

y=m;

z=k;

}

void display();

void operator-();

};

void A::operator-()

{

x=-x;

y=-y;

z=-z;

}

void A::display()

{

cout<<"\n x= "<<x;

cout<<"\n y= "<<y;

cout<<"\n z= "<<z;

}

int main()

{

A a(10,-20,30),b(20,40,-10);

cout<<"\n before negation \n";

a.display();

b.display();

-a;

-b;

cout<<"\n after negation \n";

a.display();

b.display();

}

**Output:**

before negation

x= 10

y= -20

z= 30

x= 20

y= 40

z= -10

after negation

x= -10

y= 20

z= -30

x= -20

y= -40

z= 10

**b)binary operator.**

#include<iostream>

using namespace std;

class A

{

int x,y,z;

public:

void read()

{

cout<<"\n read x,y,z \n";

cin>>x>>y>>z;

}

void display();

A operator+(A a);

};

void A::display()

{

cout<<"\n x= "<<x;

cout<<"\n y= "<<y;

cout<<"\n z= "<<z;

}

A A::operator+(A a)

{

A b;

b.x=x+a.x;

b.y=y+a.y;

b.z=z+a.z;

return b;

}

int main()

{

A a1,a2,a3;

a1.read();

a2.read();

a1.display();

a2.display();

a3=a1+a2;

a3.display();

}

**output:**

read x,y,z

10

20

30

read x, y,z

30

20

10

x= 10

y= 20

z= 30

x= 30

y= 20

z= 10

x= 40

y= 40

z= 40

**Rubrics for Evaluation of Experimental Work**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Unsatisfactory** | **Satisfactory** | **Good** | **Excellent** |
| **Observation**  **(5M)** | Completed less than 50% of the assigned programs | Completed between 50-70% of the assigned programs | Completed between 70-90% of the assigned programs | Completed between 90-100% of the assigned programs |
| **Record**  **(5M)** | * Completed less than 50% of the assigned programs * No Index * No program name, date, or Page. No included * Disorganized work | * Completed between 50-70% of the assigned programs * Index without proper order * No program name, date, or Page. No included * 50 % Disorganized work | * Completed between 70-90% of the assigned programs * Includes program name, date, and Page. No. * Organized work. | * Completed between 90-100% of the assigned programs * Includes program name, date, and Page. No. * Creatively organized work. |
| **Execution(5M)** | Does not execute due to errors. | Program executes with minor errors (easily fixed error) | * Executes without errors. * Output has minor errors | * Program executes correctly with no syntax or runtime errors |

**Unsatisfactory: 0-1 Satisfactory: 2-3 Good : 3-4 Excellent : 5**

**Direct Attainment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. No.** | **HT NO.** | **Internal (25M)** | | | **External (50M)** |
| DTD  (15M) | Exam  (10M) | Internal Total (25M) |
| 1 | 16911A1201 | 14 | 5 | 19 | 45 |
| 2 | 16911A1202 | 15 | 7 | 22 | 46 |
| 3 | 16911A1203 | 10 | 5 | 15 | 41 |
| 4 | 16911A1204 | 15 | 9 | 24 | 45 |
| 5 | 16911A1205 | 10 | 6 | 16 | 40 |
| 6 | 16911A1206 | 10 | 6 | 16 | 35 |
| 7 | 16911A1207 | 10 | 5 | 15 | 49 |
| 8 | 16911A1208 | 15 | 10 | 25 | 38 |
| 9 | 16911A1209 | 10 | 13 | 23 | 49 |
| 10 | 16911A1210 | 15 | 10 | 25 | 47 |
| 11 | 16911A1211 | 14 | 3 | 17 | 48 |
| 12 | 16911A1212 | 10 | 14 | 24 | 45 |
| 13 | 16911A1213 | 12 | 12 | 24 | 46 |
| 14 | 16911A1214 | 6 | 9 | 15 | 45 |
| 15 | 16911A1215 | 10 | 5 | 15 | 41 |
| 16 | 16911A1216 | 10 | 6 | 16 | 45 |
| 17 | 16911A1217 | 15 | 10 | 25 | 39 |
| 18 | 16911A1218 | 10 | 4 | 14 | 36 |
| 19 | 16911A1219 | 15 | 5 | 20 | 58 |
| 20 | 16911A1220 | 13 | 3 | 16 | 39 |
| 21 | 16911A1221 | 10 | 9 | 19 | 45 |
| 22 | 16911A1222 | 11 | 10 | 21 | 35 |
| 23 | 16911A1223 | 10 | 6 | 16 | 39 |
| 24 | 16911A1224 | 10 | 8 | 18 | 47 |
| 25 | 16911A1225 | 11 | 10 | 21 | 41 |
| 26 | 16911A1226 | 15 | 10 | 25 | 42 |
| 27 | 16911A1227 | 15 | 1 | 16 | -1 |
| 28 | 16911A1228 | 15 | 10 | 25 | 36 |
| 29 | 16911A1230 | 12 | 10 | 22 | 32 |
| 30 | 16911A1231 | 15 | 10 | 25 | 44 |
| 31 | 16911A1232 | 14 | 10 | 24 | 46 |
| 32 | 16911A1233 | 10 | 10 | 20 | 49 |
| 33 | 16911A1234 | 10 | 8 | 18 | 32 |
| 34 | 16911A1235 | 10 | 6 | 16 | 47 |
| 35 | 16911A1236 | 9 | 9 | 18 | 38 |
| 36 | 16911A1237 | 16 | 9 | 25 | 49 |
| 37 | 16911A1238 | 15 | 9 | 24 | 45 |
| 38 | 16911A1239 | 16 | 9 | 25 | 46 |
| 39 | 16911A1240 | 15 | 5 | 20 | 28 |
| 40 | 16911A1241 | 13 | 8 | 21 | 45 |
| 41 | 16911A1242 | 15 | 3 | 18 | 48 |
| 42 | 16911A1243 | 13 | 5 | 18 | 37 |
| 43 | 16911A1244 | 15 | 10 | 25 | 49 |
| 44 | 16911A1245 | 13 | 10 | 23 | 46 |
| 45 | 16911A1246 | 10 | 10 | 20 | 43 |
| 46 | 16911A1247 | 15 | 9 | 24 | 48 |
| 47 | 16911A1248 | 13 | 9 | 22 | 34 |
| 48 | 16911A1249 | 15 | 10 | 25 | 46 |
| 49 | 16911A1250 | 15 | 9 | 24 | 49 |
| 50 | 16911A1251 | 15 | 9 | 24 | 35 |
| 51 | 16911A1252 | 13 | 9 | 22 | 37 |
| 52 | 16911A1253 | 13 | 10 | 23 | 33 |
| 53 | 16911A1256 | 15 | 1 | 16 | 34 |
| 54 | 16911A1257 | 15 | 10 | 25 | 49 |
| 55 | 16911A1258 | 15 | 9 | 24 | 46 |
| 56 | 16911A1259 | 15 | 6 | 21 | 36 |
| 57 | 16911A1260 | 14 | 5 | 19 | 39 |
| 58 | 15911A1225 | 15 | 4 | 19 | 38 |
| 59 | 15911A1231 | 12 | 5 | 17 | 31 |
| 60 | 15911A1246 | 12 | 7 | 19 | 31 |
| No. of students attempted | | | | 60 | 59 |
| % of students scored >= 60% | | | | 98.33 | 98.31 |
| CO Direct Attainment | | | | 3 | 3 |

**Course End Survey for Indirect Attainment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **COs** | **No. of Students given 1** | **No. of Students given 2** | **No. of Students given 3** | **No. of Students participated in survey** | **Indirect Assessment** |
| **CO1** | 1 | 3 | 46 | 50 | 2.90 |
| **CO2** | 2 | 3 | 45 | 50 | 2.86 |
| **CO3** | 1 | 5 | 44 | 50 | 2.86 |
| **CO4** | 2 | 6 | 42 | 50 | 2.80 |
| **CO5** | 2 | 4 | 44 | 50 | 2.84 |
| **Average** | | | | | **2.85** |

**Course Closure Report**

**Batch :** 2016-20

**Academic Year/Semester :** 2017-18 / II semester

**Course Name :** Data Structures lab

**Course Number :** A13585

**Direct Attainment :** 3

(75% of End Exam+ 25% of Internal Exam)

**Indirect Attainment (Course End Survey) :** 2.85

**Overall Course Attainment :** 2.97

(80% of Direct + 20% of Indirect)

**Remarks:** Target Reached

**Course Coordinator**