Practical No.1

1. Write a C++ program to create a simple calculator.

Code:-

# include <iostream> using namespace std; int main() {

char op;

float num1, num2;

cout << "Enter operator: +, -, \*, /: "; cin >> op;

cout << "Enter two operands: "; cin >> num1 >> num2; switch(op) {

case '+':

cout << num1 << " + " << num2 << " = " << num1 + num2; break;

case '-':

cout << num1 << " - " << num2 << " = " << num1 - num2; break;

case '\*':

cout << num1 << " \* " << num2 << " = " << num1 \* num2; break;

case '/':

cout << num1 << " / " << num2 << " = " << num1 / num2; break;

default:

cout << "Error! operator is not correct"; break;

}

return 0;

}

Output:-

Enter operator: +, -, \*, /: + Enter two operands: 2 3

2 + 3 = 5

1. Write a C++ program to convert seconds into hours, minutes and seconds. Code:-

#include <iostream> using namespace std; int main()

{

int time = 0; int hour = 0; int min = 0; int sec = 0;

cout << "Enter a time in seconds: "; cin >> time;

hour = time/3600; time = time%3600; min = time/60; time = time%60; sec = time;

cout<<"\nThe time in HH:MM:SS format is: "<<hour<<" hours, "

<<min<<" minutes, and "<<sec<<" seconds!\n"; return 0;

}

Output:-

Enter a time in seconds: 5000

The time in HH:MM:SS format is: 1 hours, 23 minutes, and 20 seconds!

1. Write a C++ program to find the volume of a square, cone, and rectangle.
2. Code for Volume of square:- #include <iostream>

#include <string> using namespace std; int main ()

{

float s, V;

cout << "Input Square's side: "; cin >> s;

V = s\*s\*s;

cout << "The volume of the square is: " << V ; return 0;

}

Output:-

Input Square's side: 5.5

The volume of the square is: 166.375

1. Code for volume of cone:- #include <iostream>

#include <string.h> using namespace std; int main ()

{

const float pi = 3.14159; float R, H, V;

cout << "Input Cone's radius: "; cin >> R;

cout << "Input Cone's height: "; cin >> H;

V = (1.0/3.0)\*pi\*(R\*R)\*H;

cout << "The volume of the cone is: " << V ; return 0;

}

Output:-

Input Cone's radius: 6 Input Cone's height: 8

The volume of the cone is: 301.593

1. Code for volume of Rectangle:- #include <iostream>

using namespace std; int main ()

{

float L, W, H, V;

cout << "Input Rectangle's length: "; cin >> L;

cout << "Input Rectangle's height: "; cin >> H;

cout << "Input Rectangle's width: "; cin >> W;

V = L\*W\*H;

cout << "The volume of the Rectangle is: " << V ; return 0;

}

Output:-

Input Rectangle's length: 6 Input Reactangle's height: 5 Input Reactangle's width: 4 The volume of the cone is: 120

# Practical No:3A

**Aim**:Write a C++ program using classes and object Student to print the name of the student, roll\_no. Display the same.

**Program**: #include<iostream> using namespace std; class student

{

private:

char name[20]; int rollno; public:

void input(); void display();

};

void student::input()

{

cout<<"Enter Name:"; cin>>name; cout<<"Enter Roll No.:"; cin>>rollno;

}

void student::display()

{

cout<<"\nName:"<<name; cout<<"\nRoll No.:"<<rollno;

}

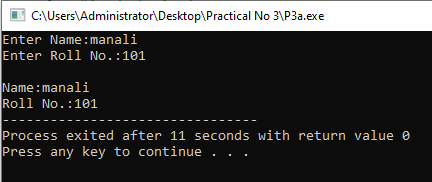
int main()

{

student s; s.input();

s.display();

}

**Output:**

# Practical No: 3B

**Aim**:Write a C++ program for Structure bank employee to print name of the employee, account\_no. & balance. Display the same also display the balance after withdraw and deposit.

**Program:** #include<iostream> #include<stdio.h> #include<string.h> using namespace std; class Bankacc

{

int acno;

char AccountHolderName[100], Account\_Type[100]; float bal;

public:

Bankacc(int acc\_no, char \*name, char \*acc\_type, float Balance) //Parameterized Constructor

{

acno=acc\_no; strcpy(AccountHolderName, name); strcpy(Account\_Type, acc\_type); bal=Balance;

}

void deposit(); void withdraw(); void Show();

};

void Bankacc::deposit() //depositing an amount

{

int DepositAmmount1;

cout<<" Enter Deposit Amount = "<<endl; cin>>DepositAmmount1; bal+=DepositAmmount1;

}

void Bankacc::withdraw() //withdrawing an amount

{

int WithdrawAmmount1;

cout<<" Enter Withdraw Amount = "<<endl; cin>>WithdrawAmmount1; if(WithdrawAmmount1>bal)

cout<<" Cannot Withdraw Amount"<<endl; bal-=WithdrawAmmount1;

}

void Bankacc::Show() //Showing the details

{

cout<<" \*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl; cout<<" Account No. : "<<acno;

cout<<" Name : "<<AccountHolderName; cout<<" Account Type : "<<Account\_Type; cout<<" Balance : "<<bal;

}

int main()

{

int acc\_no;

char name[100], acc\_type[100]; float Balance;

cout<<" Enter Details: "<<endl; cout<<"\*\*\*\*\*\*\*\*\*\*\*"<<endl; cout<<" Account No. "<<endl; cin>>acc\_no;

cout<<" Name : "<<endl; cin>>name;

cout<<" Account Type : "<<endl; cin>>acc\_type;

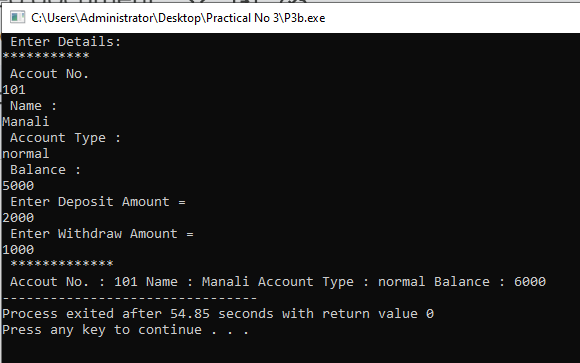
cout<<" Balance : "<<endl; cin>>Balance;

Bankacc b1(acc\_no, name, acc\_type, Balance); //object is created b1.deposit(); //

b1.withdraw(); // calling member functions b1.Show(); //

return 0;

}

**Output:**

# Practical No: 3C

**Aim**: Write a C++ Program to design a class having a static member function named showcount() which has the property of displaying the number of objects created from the class.

#### Program:

#include <iostream> using namespace std; class test {

int objNo;

static int objCnt;

public:

test()

{

objNo = ++objCnt;

}

~test()

{

--objCnt;

}

void printObjNumber(void)

{

cout << "object number :" << objNo << "\n";

}

static void printObjCount(void)

{

cout << "count:" << objCnt<< "\n";

}

};

int test::objCnt; int main()

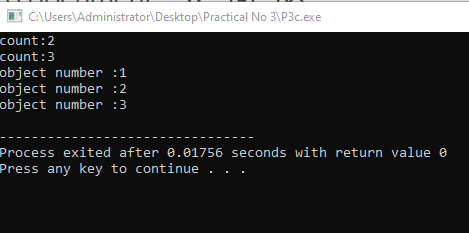
{

test t1, t2; test::printObjCount();

test t3; test::printObjCount();

t1.printObjNumber(); t2.printObjNumber(); t3.printObjNumber(); return 0;

}

**Output**:

# Practical No :3D

**Aim:**Write a Program to find Maximum out of Two Numbers using friend function. Note: Here one number is a member of one class and the other number is a member of some other class.

**Program:** #include<iostream> using namespace std; class a;

class b

{

int number; public:

b(int x) // parameterized constructor

{

number=x;

}

void friend greatest(a a1,b b1);

};

class a

{

int number; public:

a(int x) // parameterized constructor

{

number=x;

}

void friend greatest(a a1,b b1);

};

void greatest(a a1,b b1)

{

if(a1.number>b1.number)

{

cout<<"\n Number in class A is greatest i.e."<<a1.number;

}

else if(a1.number<b1.number)

{

cout<<"\n Number in class B is greatest i.e."<<b1.number;

}

else

{

cout<<"\n Number in both classes are equal";

}

}

int main()

{

cout<<"\n\n Program to find greatest of two numbers in two different classes using friend function";

int num;

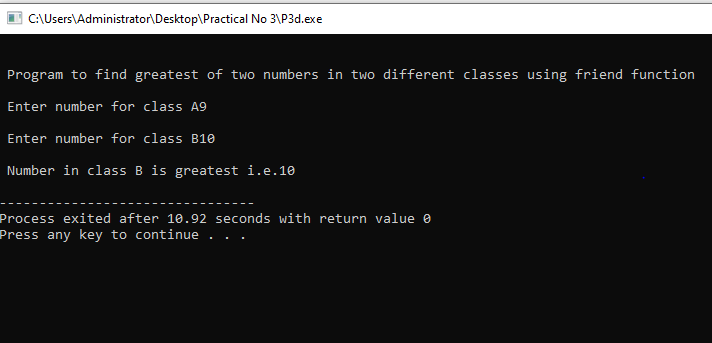
cout<<"\n\n Enter number for class A" ; cin>>num;

a a1(num);

cout<<"\n Enter number for class B"; cin>>num;

b b1(num); greatest(a1,b1); cout<<"\n"; return 0;

}

**Output:**

# Practical No: 3E

**Aim:**Write a C++ Program using copy constructor to copy data of an object to another object.

#### Program:

#include <iostream> using namespace std; class A

{

public:

int x;

A(int a) // parameterized constructor.

{

x=a;

}

A(A &i) // copy constructor

{

x = i.x;

}

};

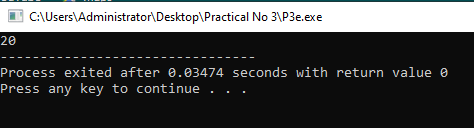
int main()

{

A a1(20); // Calling the parameterized constructor. A a2(a1); // Calling the copy constructor. cout<<a2.x;

return 0;

}

**Output:**

# Practical No:3F

Aim:Write a C++ Program to allocate memory dynamically for an object of a given class using the class's constructor.

Program

#include <iostream> using namespace std;

class abc { const char\* p;

public:

// default constructor abc()

{

// allocating memory at run time p = new char[6];

p = "FYIT";

}

void display() { cout << p << endl; }

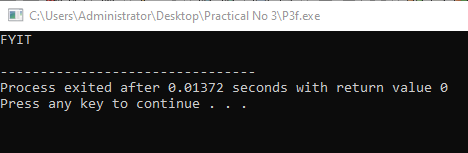
};

int main()

{

abc obj; obj.display();

}

**Output:**

**Practical No 4A**

**Aim:** Write a C++ program to illustrate single inheritance.

**Program:** #include<iostream> #include<conio.h> using namespace std; class person

{

int age;

char name[30]; public:

void getdata()

{

cout<<"\n enter name and age:"; cin>>name;

cin>>age;

}

void show()

{

cout<<"\n name="<<name; cout<<"\n age="<<age;

}

};

class employee:public person

{

float salary; public:

void getsal()

{

cout<<"\n enter salary:"; cin>>salary;

}

void showsal()

{

cout<<"\n salary:"<<"\n"<<salary;

}

};

int main()

{

employee e;

cout<<"\n enter employee data"; e.getdata();

e.getsal();

cout<<"\n employee info"; e.show();

e.showsal();

getch();

}

#### Output:-

enter employee data

enter name and age:Anshu 25 enter salary:25000

employee info name =Anshu age=25 salary= 25000

## Practical No 4B

**Aim:** Write a C++ Program that illustrates multiple inheritance.

**Code:-** #include<iostream> #include<conio.h> using namespace std;

class person // base class1

{

char name[20]; int age;

public:

void getdata()

{

cout<<"Enter the name and age:"; cin>>name>>age;

}

void show()

{

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

}

};

class employee // base class 2

{

double salary; public:

void getsal()

{

cout<<"\n Enter your salary:"; cin>>salary;

}

void showsal()

{

cout<<"\n Salary is:"<<salary;

}

};

class fulltime:public person,public employee

{

float time; public:

void gettime()

{

cout<<"\n Enter the time:"; cin>>time;

}

void showtime()

{

cout<<"\n time is:"<<time;

}

};

int main()

{

fulltime f;

cout<<"\n enter employee data:\n"; f.getdata();

f.getsal();

f.gettime();

cout<<"\n employee information:"; f.show();

f.showsal();

f.showtime();

getch();

}

#### Output:-

enter employee data:

Enter the name and age:sagar 26 Enter your salary:27000

Enter the time:2.30

employee information: name:=sagar

age:=26

Salary is:=27000 time is:2.3

## Practical No 4C

**Aim:** Write a C++ Program that illustrate multi level inheritance.

#### Code:-

#include <iostream> using namespace std;

class Vehicle //base class1;

{

public:

void vehicle()

{

cout<<”I am a vehicle\n”;

}

};

class FourWheeler : public Vehicle

{

public:

void fourWheeler()

{

cout<<”I have four wheels\n”;

}

};

class Car : public FourWheeler

{

public:

void car()

{

cout<<”I am a car\n”;

}

};

int main()

{

Car obj; obj.car();

obj.fourWheeler();

obj.vehicle(); return 0;

}

#### Output:-

I am a car

I have four wheels I am a vehicle

## Practical No 4D

**Aim:**Write a C++ Program that illustrates Hierarchical inheritance.

#### Program:

// C++ program to illustrate the above concept #include <iostream>

#### using namespace std;

// Base class **class** shape { **public**:

string name;

**int** sides;

shape(string name, **int** sides) // constructor

{

**this**->name = name; // this pointer

**this**->sides = sides;

}

};

// Derived class

**class** triangle : **public** shape // mode is public

{

#### private:

**int** base;

**int** height;

#### public:

// shape constructor taking arguments

// from triangle constructor

triangle(string name, **int** sides, **int** base, **int** height) : shape(name, sides)

{

**this**->base = base;

**this**->height = height;

}

**void** area()

{

int area=0.5\*base\*height; cout << "area of triangle: "

<< area<< endl;

}

**void** details()

{

cout << "shape is: " << name << endl;

cout << "no. of sides are: " << sides << endl; cout << "base is: " << base << endl;

cout << "height is: " << height << endl; area(); // calling area()

}

};

// Derived class

**class** square : **public** shape {

#### private:

**int** height;

#### public:

// shape constructor taking arguments

// from square constructor

square(string name, **int** sides, **int** height) : shape(name, sides)

{

**this**->height = height;

}

**void** area()

{

cout << "area of square: " << (height \* height)

<< endl;

}

**void** details()

{

cout << "shape is: " << name << endl;

cout << "no. of sides are: " << sides << endl; cout << "height is: " << height << endl; area(); // calling area()

}

};

**int** main()

{

// Creating objects

triangle t("triangle", 3, 2, 3);

square s("square", 4, 2);

t.details();

cout << endl << endl;

s.details();

**return** 0;

}

#### Output:

shape is: triangle no. of sides are: 3 base is: 2

height is: 3

area of triangle: 3

shape is: square no. of sides are: 4 height is: 2

area of square: 4

## Practical No 4E

**Aim:** Write a C++ Program to generate Fibonacci Series by using Constructor to initialize the Data Members.

#### Program:

#include<iostream> using namespace std; class fibonacci

{

public:

fibonacci(int num) // parameterized constructor

{

int a=0,b=1,t,j; cout<<"enter the number :"; cin>>num;

cout<<"the fibonacci series is :"<<a<<b; for(j=1;j<=num;j++)

{

t=a+b; cout<<t; a=b; b=t;

}

cout<<"\n";

}

};

int main()

{

fibonacci obj(5);

}

#### Output:

enter the number :4

the fibonacci series is :011235

## Practical No 4F

**Aim:**Write a C++ program to overload new/delete operators in a class.

**Program:** #include<iostream> #include<stdlib.h>

using namespace std; class student

{

public:

string name; int age;

student() // default constructor

{

cout<< "Constructor is called\n" ;

}

student(string name, int age) // parameterized constructor

{

this->name = name; this->age = age;

}

void display()

{

cout<< "Name:" << name << endl; cout<< "Age:" << age << endl;

}

void \* operator new(size\_t size)

{

cout<< "Overloading new operator with size: " << size << endl; void \* p = ::operator new(size);

//void \* p = malloc(size); //will also work fine

return p;

}

void operator delete(void \* p)

{

cout<< "Overloading delete operator " << endl; free(p);

}

};

int main()

{

student \* p = new student("Yash", 24);

p->display(); delete p;

}

#### Output:

Overloading new operator with size: 40 Name:Yash

Age:24

Overloading delete operator

# Practical No. 05

**Aim:** Write a C++ Program to design a student class representing student roll no. and a test class (derived class of student) representing the scores of the student in various subjects and sports class representing the score in sports. The sports and test class should be inherited by a result class having the functionality to add the scores and display the final result for a student.

**Code:-** #include<iostream> using namespace std; class student

{

protected:

int roll\_number; public:

void get\_number(int a)

{

roll\_number = a;

}

void put\_number(void)

{

cout<<"Roll No:"<<roll\_number<<"\n";

}

};

class test : public student

{

protected:

float OOP, WAD; public:

void get\_marks(float x, float y)

{

OOP= x;

WAD= y;

}

void put\_marks(void)

{

cout<<"Marks obtained"<<"\n"<<"OOP ="<<OOP<<"\n"<<"WAD ="<<WAD<<"\n";

}

};

class sports

{

protected:

float score; public:

void get\_score(float s)

{

score = s;

}

void put\_score(void)

{

cout<<"Sports wt:"<<score<<"\n\n";

}

};

class result : public test, public sports

{

float total; public:

void display(void);

};

void result ::display(void)

{

total = OOP + WAD + score; put\_number();

put\_marks(); put\_score();

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

}

int main()

{

result student\_1; student\_1.get\_number (1234);

student\_1.get\_marks (27.5, 33.0);

student\_1.get\_score (6.0); student\_1.display (); return 0;

}

Output:-

Roll No:1234 Marks obtained part1 =27.5

part2 =33 Sports wt:6

Total Score:66.5

Practical No. 06

A) Write a C++ program to maintain the records of a person with details (Name and Age) and find the eldest among them. The program must use this pointer to return the result.

Code:- #include<iostream> #include<stdio.h> #include<conio.h> using namespace std; class Person

{

char Name[20]; int Age;

public:

void getData()

{

cout<<"Enter Person Name:"; cin>>Name;

cout<<"Enter Person Age:"; cin>>Age;

}

void putData()

{

cout<<"\n\nDetails About Eldest Person as follows:"; cout<<"\nName="<<Name<<"\nAge="<<Age;

}

Person &Compare(Person &p1)

{

if(p1.Age>this->Age) return p1;

return \*this;

}

};

int main()

{

Person x,y,z; x.getData();

y.getData(); z=x.Compare(y); z.putData();

getch();

}

Output:-

Enter Person Name:Sagar Enter Person Age:40

Enter Person Name:Pushpa Enter Person Age:30

Details About Eldest Person as follows: Name=Sagar

Age=40

Practical No.7

1. Write a C++ program illustrating the use of virtual functions in class. Code:-

#include<iostream> using namespace std; class base

{

public:

virtual void print()

{

cout << "print base class\n";

}

void show()

{

cout << "show base class\n";

}

};

class derived : public base

{

public:

void print()

{

cout << "print derived class\n";

}

void show()

{

cout << "show derived class\n";

}

};

int main()

{

base \*bptr; derived d; bptr = &d; bptr->print(); bptr->show(); return 0;

}

Output:-

print derived class show base class

1. Write a C++ program to design a class representing the information regarding digital library (books, tape: book & tape should be separate classes having the base class as media). The class should have the functionality for adding new item, issuing, deposit etc. the program should use the runtime polymorphism.

Code:- #include<iostream> #include<string.h> using namespace std; class media

{

protected:

char title[50]; float price; public:

media(char \*s, float a)

{

strcpy(title, s); price = a;

}

virtual void display(){}

};

class book : public media

{

int pages; public:

book(char \*s, float a, int p) : media(s,a)

{

pages = p;

}

void display();

};

class tape : public media

{

float time; public:

tape(char \* s, float a, float t):media(s,a)

{

time =t;

}

void display();

};

void book ::display()

{

cout<<"\n Title:"<<title; cout<<"\n Pages:"<<pages; cout<<"\n Price:"<<price;

}

void tape ::display ()

{

cout<<"\n Title:"<<title;

cout<<"\n Play Time:"<<time<<"mins"; cout<<"\n Price:"<<price;

}

int main()

{

char \* title = new char[30]; float price, time;

int pages;

cout<<"\n Enter Book Details \n"; cout<<"\n Title:";

cin>>title; cout<<"\n Price:"; cin>>price; cout<<"\n Pages:"; cin>>pages;

book book1(title, price, pages); cout<<"\n Enter Tape Details"; cout<<"\n Title:";

cin>>title; cout<<"\n Price:"; cin>>price;

cout<<"\n Play Times(mins):"; cin>>time;

tape tape1(title, price, time); media\* list[2];

list[0] = &book1; list[1] = &tape1;

cout<<"\n Media Details"; cout<<"\n..............Book. ";

list[0]->display (); cout<<"\n..............Tape. ";

list[1]->display (); return 0;

}

Output:-

Enter Book Details Title:OOP Price:185 Pages:60

Enter Tape Details Title:OOP Price:200

Play Times(mins):60

Media Details

..............Book.....

Title:OOP Pages:60 Price:185

..............Tape.....

Title:OOP

Play Time:60mins Price:200

**Practical No: 9A**

**Aim:** Write a C++ program to show conversion from string to int.

#### Source Code:

// C++ program to demonstrate working of stoi() Work only if compiler supports C++11 or above. Because STOI()was added in C++ after 2011

#include <iostream> #include <string> using namespace std;

// Driver code int main()

{

string str1 = "45"; string str2 = "3.14159";

char str3[] = "31337 geek";

// type of explicit type casting int myint1 = stoi(str1);

// type of explicit type casting int myint2 = stoi(str2);

// type of explicit type casting int myint3 = stoi(str3);

cout << "stoi(\"" << str1 <<

"\") is " << myint1 << '\n'; cout << "stoi(\"" << str2 <<

"\") is " << myint2 << '\n'; cout << "stoi(\"" << str3 <<

"\") is " << myint3 << '\n';

return 0;

}

#### Output:

stoi("45") is 45

stoi("3.14159") is 3 stoi("31337 geek") is 31337

### Practical No 9B

**Aim:** Write a C++ program to show conversion from int to string.

#### Source Code:

// C++ code to demonstrate "to\_string()" method to convert number to string. #include <iostream>

#include <string> // for string and to\_string() using namespace std;

// Driver Code int main()

{

// Declaring integer int i\_val = 20;

// Declaring float float f\_val = 30.50;

// Conversion of int into string using to\_string() string stri = to\_string(i\_val);

// Conversion of float into string using to\_string() string strf = to\_string(f\_val);

// Displaying the converted strings cout << "The integer in string is : "; cout << stri << endl;

cout << "The float in string is : "; cout << strf << endl;

return 0;

}

#### Output:

The integer in string is : 20

The float in string is : 30.500000

### Practical No :9C

**Aim:**Write a C++ program implementing basic operation of class ios i.e. setf, unsetf, precision etc.

#### Source Code:

#include<bits/stdc++.h> using namespace std;

// The width() function defines width of the next value to be displayed in the output at the

// console.

void IOS\_width()

{

cout << " \n";

cout << "Implementing ios::width\n\n"; char c = 'A';

// Adjusting width will be 5. cout.width(5);

cout << c <<"\n";

int temp = 10;

// Width of the next value to be displayed in the output will not be adjusted to 5 columns. cout<<temp;

cout << "\n \n";

}

void IOS\_precision()

{

cout << "\n \n";

cout << "Implementing ios::precision\n\n"; cout << "Implementing ios::width"; cout.setf(ios::fixed, ios::floatfield); cout.precision(2);

cout<<3.1422;

cout << "\n \n";

}

// The fill() function fills the unused white spaces in a value (to be printed

// at the console), with a character of choice.

void IOS\_fill()

{

cout << "\n \n"; cout << "Implementing ios::fill\n\n"; char ch = 'a';

// Calling the fill function to fill the white spaces in a value with a character our of

// choice. cout.fill('\*');

cout.width(10); cout<<ch <<"\n";

int i = 1;

// Once you call the fill() function, you don't have to call it again to

// fill the white space in a value with the same character. cout.width(5);

cout<<i;

cout << "\n \n";

}

void IOS\_setf()

{

cout << "\n \n"; cout << "Implementing ios::setf\n\n"; int val1=100,val2=200; cout.setf(ios::showpos); cout<<val1<<" "<<val2;

cout << "\n \n";

}

void IOS\_unsetf()

{

cout << "\n \n";

cout << "Implementing ios::unsetf\n\n"; cout.setf(ios::showpos|ios::showpoint);

// Clear the showflag flag without affecting the showpoint flag cout.unsetf(ios::showpos);

cout<<200.0;

cout << "\n \n";

}

// Driver Method int main()

{

IOS\_width(); IOS\_precision(); IOS\_fill(); IOS\_setf(); IOS\_unsetf(); return 0;

}

#### Output:

Implementing ios::width

A 10

Implementing ios::fill

\*\*\*\*\*\*\*\*\*a

\*\*\*\*1

Implementing ios::setf

+100 +200

Implementing ios::unsetf

200.000

### Practical No:9D

**Aim:**Write a C++ program to implement I/O operations on characters.

**Source Code:** #include <iostream> using namespace std;

int main()

{

char ch;

cout<<"Press any key: "; ch = cin.get();

cout << "You have pressed: "; cout.put(ch);

return 0;

}

#### Output:

Press any key: sai You have pressed: s

### Practical No: 9E

**Aim:**Write a C++ program to copy the contents of one file to another.

#### Source Code:

// C++ Program to demonstrate

// copying the content of a .txt file #include <fstream>

#include <iostream> using namespace std;

int main()

{

string line;

// For writing text file creating ofstream & ifstream class object ifstream ini\_file{

"original.txt"

}; // This is the original file ofstream out\_file{ "copy.txt" }; if (ini\_file && out\_file) {

while (getline(ini\_file, line)) { out\_file << line << "\n";

}

else {

}

}

cout << "Copy Finished \n";

// Something went wrong printf("Cannot read File");

// Closing file ini\_file.close(); out\_file.close(); return 0;

}

#### Output:

Cannot read File

**Practical No.10**

1. Write a C++ program to create Simple calculator using class template. Code:-

#include <iostream> using namespace std; template <class T> class Calculator

{

private:

T num1, num2; public:

Calculator(T n1, T n2)

{

num1 = n1; num2 = n2;

}

void displayResult()

{

cout << "Numbers are: " << num1 << " and " << num2 << "." << endl; cout << "Addition is: " << add() << endl;

cout << "Subtraction is: " << subtract() << endl; cout << "Product is: " << multiply() << endl; cout << "Division is: " << divide() << endl;

}

T add() { return num1 + num2; }

T subtract() { return num1 - num2; } T multiply() { return num1 \* num2; } T divide() { return num1 / num2; }

};

int main()

{

Calculator<int> intCalc(2, 1);

Calculator<float> floatCalc(2.4, 1.2); cout << "Int results:" << endl; intCalc.displayResult();

cout << endl << "Float results:" << endl; floatCalc.displayResult();

return 0;

}

Output:- Int results:

Numbers are: 2 and 1.

Addition is: 3

Subtraction is: 1

Product is: 2

Division is: 2

Float results:

Numbers are: 2.4 and 1.2.

Addition is: 3.6

Subtraction is: 1.2

Product is: 2.88

Division is: 2

1. Write a C++ program to get maximum of two number using class template. Code:-

#include<iostream> #include<stdio.h> #include<conio.h> using namespace std; template<class T> class TClassMax {

T x, y; public:

TClassMax() { } TClassMax(T first, T second) {

x = first;

y = second;

}

T getMaximun() { if (x > y)

return x; else

return y;

}

};

int main()

{

TClassMax <int> iMax; // (100, 75); int a, b, i;

TClassMax <float> fMax; // (90.78, 750.98); float c, d, j;

cout << "Class Template Programs : Generic Programming : Get Maximum Number \n"; cout << "Enter A,B values(integer):";

cin >> a>>b;

iMax = TClassMax<int>(a, b); i = iMax.getMaximun();

cout << "Result Max Int : " << i;

cout << "\n\nEnter C,D values(float):";

cin >> c>>d;

fMax = TClassMax<float>(c, d); j = fMax.getMaximun();

cout << "Result Max Float : " << j; getch();

return 0;

}

Output:-

Class Template Programs : Generic Programming : Get Maximum Number Enter A,B values(integer):20

10

Result Max Int : 20

Enter C,D values(float):20.3 30.8

Result Max Float : 30.8

Practical No: 2

#### Aim: Write a C++ program to find the sum of even and odd n natural numbers.

**Program**:

# include <iostream> using namespace std; int main()

{

int n, oddSum=0, evenSum=0; cout<<"\n Enter a number:"; cin>>n;

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

{

if (i % 2 == 0)

evenSum = evenSum + i; else

oddSum = oddSum + i;

}

cout<<"Sum of odd numbers:"<<oddSum<<"\n"; cout<<"Sum of even numbers:"<<evenSum; cout<<endl;

return 0;

}

#### Output:

Enter a number:20

Sum of odd numbers:100 Sum of even numbers:110

#### Aim: Write a C++ program to find the greatest of three numbers.

**Program**:

# include <iostream> using namespace std; int main()

{

int a, b, c;

cout<<"Enter three number:"; cin>>a>>b>>c;

if (a>=b && a>=c)

{

cout<<"Greatest number is:"<<a;

}

if (b>=c && b>=c)

{

cout<<"Greatest number is:"<<b;

}

if (c>=a && c>=b)

{

cout<<"Greatest number is:"<<c;

}

return 0;

}

#### Output:

Enter three number:34 56 23 Greatest number is:56

#### Aim: Write a C++ program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

**Program**:

#include <iostream> using namespace std;

// Function to check if a given number is prime bool isPrime(int n)

{

// Since 0 and 1 is not prime return false. if(n == 1 || n == 0) return false;

// Run a loop from 2 to n-1 for(int i = 2; i < n; i++)

{

// if the number is divisible by i,

// then n is not a prime number. if(n % i == 0) return false;

}

// Otherwise n is a prime number. return true;

}

// Driver code int main()

{

int N;

cout<<"enter number:"; cin>>N;

// Check for every number from 1 to N for(int i = 1; i <= N; i++)

{

// Check if current number is prime if(isPrime(i))

{

cout << i << " ";

}

}

return 0;

}

#### Output:

enter number:10 2 3 5 7