**CPP ASSIGNMENT**

**Kshitija S. Kurhade.**

**243160**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Lab 1**

1:write program to test Hello World.

#include<iostream>

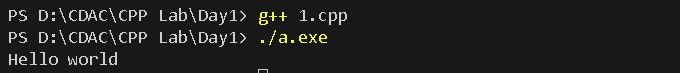
using namespace std;

int main(){

cout<<"Hello world";

return 0;

}

****

2:Write a program to adddition of two numbers .

#include<iostream>

using namespace std;

int main(){

int a,b,c;

cout<<"Enter value of a & b :"<<endl;

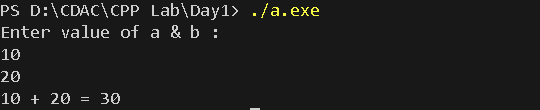
cin>>a>>b;

c = a+b;

cout<<a<<" + "<<b<<" = "<<c;

return 0;

}



3:Write a program to swap two numbers.

#include<iostream>

using namespace std;

class Swap{

public : int num1,num2;

void acceptNum(){

cout<<"Enter Num1 and Num2 :"<<endl;

cin>>num1>>num2;

}

void swapNum(){

int temp;

temp = num1;

num1 = num2;

num2 = temp;

cout<<"After Swap : "<<endl;

cout<<"num1 : "<<num1<<" num2 : "<<num2;

}

};

int main(){

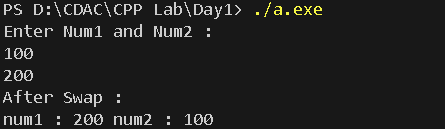
Swap obj;

obj.acceptNum();

obj.swapNum();

return 0;

}



4. Write a program to accept an integer and check if it is even or odd.

#include<iostream>

using namespace std;

int main(){

int num;

cout<<"Enter Number : "<<endl;

cin>>num;

if(num%2==0){

cout<<num<<" is even"<<endl;

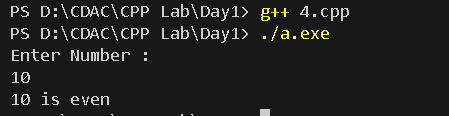
}else{

cout<<num<<" is odd"<<endl;

}

return 0;

}



5. Write a program to accept a number and check if it is divisible by 5&7.

#include<iostream>

using namespace std;

int main(){

int num;

cout<<"Enter Number : "<<endl;

cin>>num;

if(num%5==0 && num%7==0){

cout<<num<<" divisble by both 5 and 7";

}else{

cout<<num<<" not divisible by both5 and 7";

}

return 0;

}



6. Write a program, which accepts annual basic salary of an employee and calculates and displays the

Income tax as per the following rules.

Basic: < 1, 50,000 Tax = 0

1, 50,000 to 3,00,000 Tax = 20%

> 3,00,000 Tax = 30%

#include<iostream>

using namespace std;

class Employee{

private : int salary; double tax;

public :

Employee(int salary){

this->salary = salary;

}

void calcTax(){

if(salary<150000){

tax = 0;

cout<<"Tax : "<<tax;

}

else if(salary>150000 && salary<=300000){

tax = 0.2 \* salary;

cout<<"Tax : "<<tax;

}

else{

tax = 0.3 \* salary;

cout<<"Tax : "<<tax;

}

}

};

int main(){

Employee obj(300000);

obj.calcTax();

return 0;

}



7. Accept a lowercase character from the user and check whether the character is a vowel or consonant.

(Hint: a, e, i, o, u are vowels)

#include<iostream>

using namespace std;

int main(){

char ch ;

cout<<"Enter Char : "<<endl;

cin>>ch;

if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u'){

cout<<ch<<" is vowel"<<endl;

}

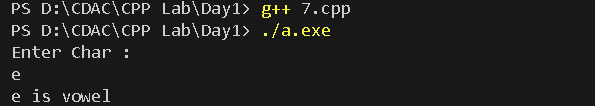
else{

cout<<ch<<" is consonant"<<endl;

}

return 0;

}



8. Write a program to input angles of a triangle and check whether triangle is valid or not.

/\* Write a program to input angles of a triangle and check whether triangle is valid or not.\*/

#include<iostream>

using namespace std;

int main(){

int angle1,angle2,angle3,res;

cout<<"Enter angle1,angle2,angle3 "<<endl;

cin>>angle1>>angle2>>angle3;

res = angle1+angle2+angle3;

if(res==180){

cout<<"Triangle is valid";

}

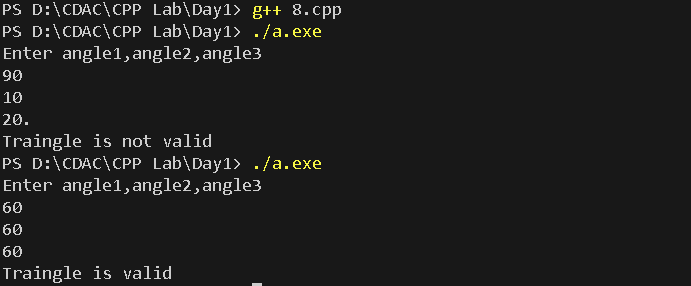
else{

cout<<"Triangle is not valid";

}

return 0;

}



9:Write a program to find factorial of a given number. ex:no5 fact=5\*4\*3\*2\*1=120.

#include<iostream>

using namespace std;

int main(){

int n, fact=1;

cout<<"Enter Number : "<<endl;

cin>>n;

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

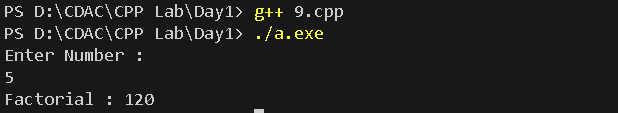
fact = fact\*i;

}

cout<<"Factorial : "<<fact;

return 0;

}



10:Write a program to find m to the power n. m=3 and n=4 so 3\*3\*3\*3.

#include<iostream>

using namespace std;

int main(){

int m, n, result=1;

cout<<"\_\_\_\_\_\_\_\_Find m to the power n \_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"Enter m and n :"<<endl;

cin>>m>>n;

int i=1;

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

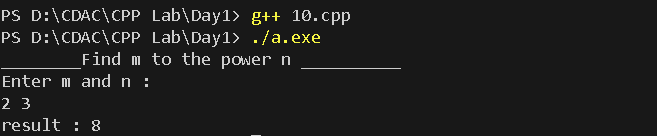
result \*= m;

}

cout<<"result : "<<result;

return 0;

}



11:Check if number is a prime number or not.:

#include<iostream>

using namespace std;

int main(){

int num, flag=0;

cout<<"Enter Number : "<<endl;

cin>>num;

for(int i = 2; i <= num/2; i++)

{

if(num%i == 0){

flag = 1;

}

}

if(flag==0){

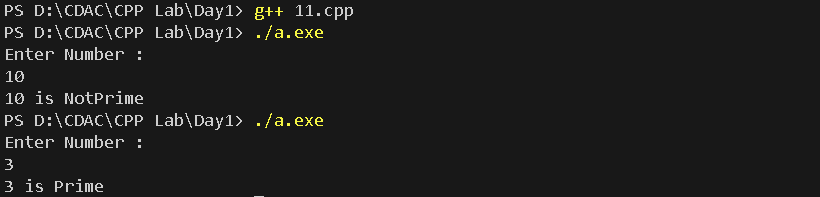
cout<<num<<" is Prime"<<endl;

}else{

cout<<num<<" is NotPrime"<<endl;

}

}



12:Sum of series :

1+2+3+….+n

#include<iostream>

using namespace std;

int main(){

int n,sum=0;

cout<<"Enter no. of elements of series"<<endl;

cin>>n;

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

{

sum+=i;

cout<<i;

if(i<n)

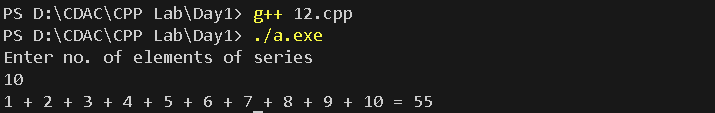
cout<<" + ";

}

cout<<" = "<<sum<<endl;

return 0 ;

}



13:Check whether the number is palindrome or not?

/\*13:Check whether the number is palindrome or not?\*/

#include<iostream>

using namespace std;

int main(){

int number, rev\_number=0;

cout<<"Enter Number : "<<endl;

cin>>number;

int org\_number = number;

while (number >0)

{

int rem = number % 10;

rev\_number = rev\_number \* 10 + rem;

number /= 10;

}

if(org\_number == rev\_number){

cout<<org\_number<<" is Palindrome";

}

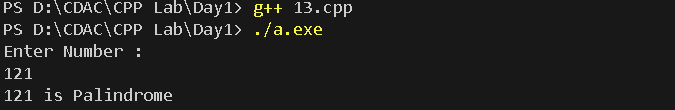
else{

cout<<org\_number<<" is Not Palindrome";

}

return 0;

}



14:Write a program to find sum of all even and odd numbers between 1 to n.

#include<iostream>

using namespace std;

int main(){

int n,evenSum=0, oddSum=0;

cout<<"Enter N : "<<endl;

cin>>n;

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

{

if(i%2==0){

evenSum += i;

}else{

oddSum += i;

}

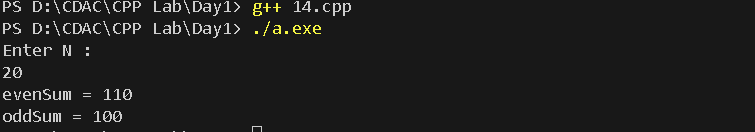
}

cout<<"evenSum = "<<evenSum<<endl;

cout<<"oddSum = "<<oddSum;

return 0;

}



15: Write a program to enter a number and print its reverse.

#include<iostream>

using namespace std;

int main(){

int number,rev\_num=0;

cout<<"enter number : "<<endl;

cin>>number;

//int org\_num = number;

while(number >0){

int rem = number % 10;

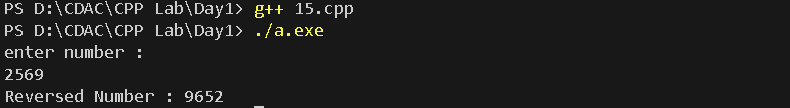
rev\_num = rev\_num \* 10 + rem;

number /=10;

}

cout<<"Reversed Number : "<<rev\_num;

}



16:Write a program to print all Prime numbers between 1 to n.

#include<iostream>

using namespace std;

int main(){

int n ;

cout<<"Enter N :"<<endl;

cin>>n;

for (int i = 2; i <= n; i++)

{

int flag =0;

for (int j = 2; j <= i/2; ++j)

{

if(i%j == 0){

flag = 1;

break;

}

}

if(flag==0){

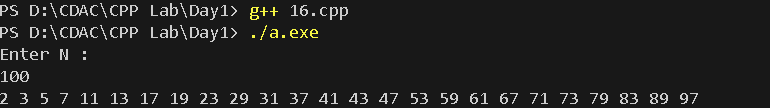
cout<<i<<" ";

}

}

return 0;

}



17:Write a program to check entered number is Armstrong number or not.

#include<iostream>

using namespace std;

int main(){

int num, rem, sum=0, temp;

cout<<"Enter Number : "<<endl;

cin>>num;

for(temp = num; num!=0; num = num/10){

rem = num % 10;

sum = sum +(rem\*rem\*rem);

}

if(sum==temp){

cout<<temp<<" is Armstromg No.";

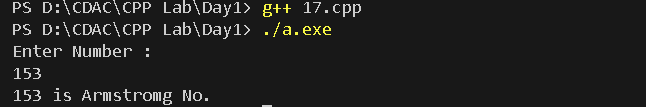
}else{

cout<<temp<<" is Not Armstromg No.";

}

return 0;

}



18:Write a program to find greatest of three numbers using nested if-else.

#include<iostream>

using namespace std;

int main(){

int a, b, c;

cout<<"Enter a b c : "<<endl;

cin>>a>>b>>c;

if(a>b){

if(a>c){

cout<<a<<" A is greater"<<endl;

}

else{

cout<<c<<" C is greater"<<endl;

}

}else{

cout<<b<<" B is greater"<<endl;

}

}



19:Create menu driven program for Pizza Shop.And display total amount.

/\*19:Create menu driven program for Pizza Shop.And display total amount\*/

#include<iostream>

using namespace std;

class pizza{

int amount;

public :

pizza(){

amount = 0;

}

void addMiniPizza(){

amount += 250;

}

void addGarlicBread(){

amount += 120;

}

void addChocoLavaCake(){

amount += 90;

}

void addPepsi(){

amount += 30;

}

void displayBill(){

cout<<"Total Amount : "<<amount<<endl;

}

};

int main(){

pizza obj;

int ch;

do{

cout<<"\_\_\_\_\_\_\_\_\_\_\_MENU\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"1.Mini Pizza \n2.Garlic Bread \n3.Choco LavaCake \n4.Pepsi \n5.Total Bill \n6.Exit \n\nEnter Choice : ";

cin>>ch;

switch(ch){

case 1 : obj.addMiniPizza();break;

case 2 : obj.addGarlicBread();break;

case 3 : obj.addChocoLavaCake();break;

case 4 : obj.addPepsi();break;

case 5 : obj.displayBill();break;

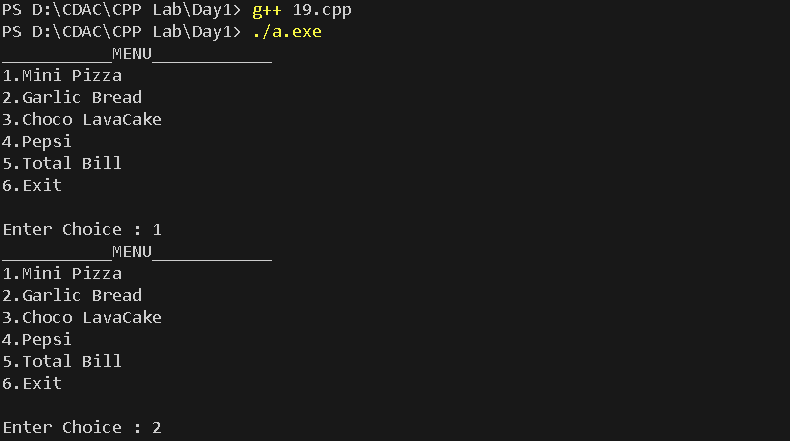
case 6 : cout<<"\nExiting....... Thank you!";break;

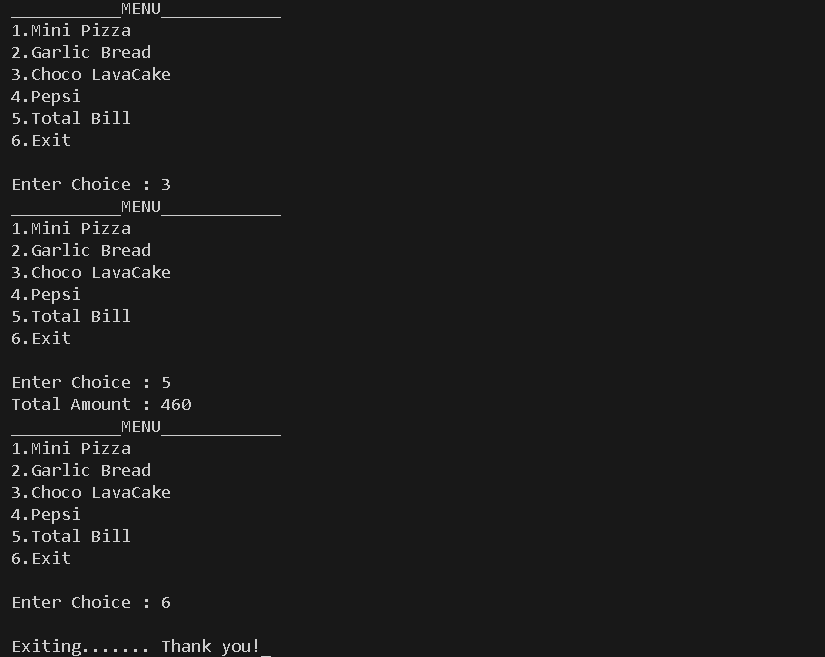
}

}while(ch!=6);

return 0;

}





20:Accept a single digit from the user and display it in words. For example, if digit entered is 9, display Nine.

#include<iostream>

using namespace std;

int main(){

int num;

cout<<"Enter Number : "<<endl;

cin>>num;

switch (num)

{

case 0: cout<<"ZERO"<<endl;break;

case 1: cout<<"ONE"<<endl;break;

case 2: cout<<"TWO"<<endl;break;

case 3: cout<<"THREE"<<endl;break;

case 4: cout<<"FOUR"<<endl;break;

case 5: cout<<"FIVE"<<endl;break;

case 6: cout<<"SIX"<<endl;break;

case 7: cout<<"SEVEN"<<endl;break;

case 8: cout<<"EIGHT"<<endl;break;

case 9: cout<<"NINE"<<endl;break;

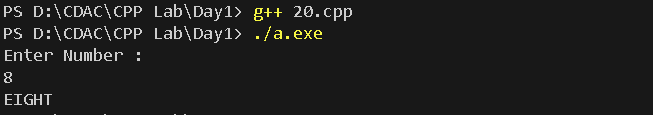
default: cout<<"Invalid Number..";

break;

}

return 0;

}



21. Write a program, which accepts two integers and an operator as a character (+ - \* / ), performs the

corresponding operation and displays the result.

/\*Write a program, which accepts two integers and an operator as a character (+ - \* / ), performs the

corresponding operation and displays the result.\*/

#include<iostream>

using namespace std;

int main(){

int num1, num2, res;

char ch;

cout<<"Enter NUM1 NUM2 : "<<endl;

cin>>num1>>num2;

cout<<"Enter Operator (+ - \* /) : "<<endl;

cin>>ch;

switch(ch){

case '+' : res = num1 + num2; cout<<num1<<" + "<<num2<<" = "<<res<<endl; break;

case '-' : res = num1 - num2; cout<<num1<<" - "<<num2<<" = "<<res<<endl; break;

case '\*' : res = num1 \* num2; cout<<num1<<" \* "<<num2<<" = "<<res<<endl; break;

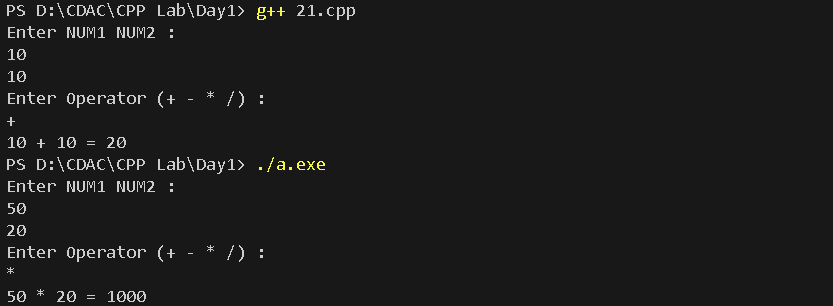
case '/' : res = num1 / num2; cout<<num1<<" / "<<num2<<" = "<<res<<endl; break;

default : cout<<"Invalid Operator!"<<endl;break;

}

return 0;

}



**Lab 2**

1:Write a program that accepts numbers continuously as long as the number is positive and prints the sum of the given numbers.

#include<iostream>

using namespace std;

int main(){

int no,sum=0;

cout<<"Enter Number : "<<endl;

while(no>0){

cin>>no;

if(no>0)

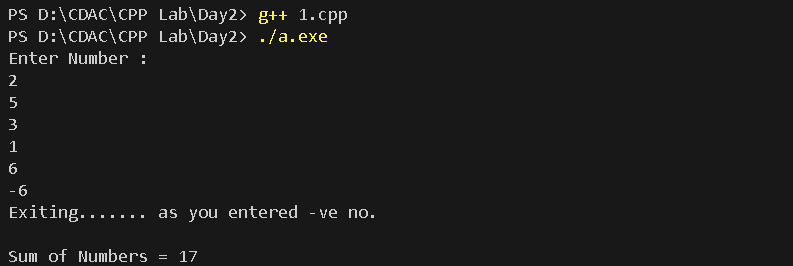
sum = sum+no;

}

cout<<"Exiting....... as you entered -ve no. \n\nSum of Numbers = "<<sum;

return 0;

}



2. Write a program to accept two integers x and n and compute x raised to n.

#include<iostream>

using namespace std;

int main(){

int x,n,res=1, i=1;

cout<<"Enter X and N : "<<endl;

cin>>x>>n;

while(i<=n){

res \*= x;

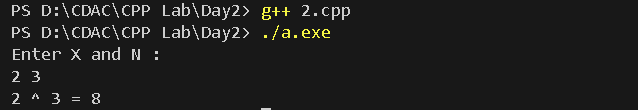
i++;

}

cout<<x<<" ^ "<<n<<" = "<<res;

return 0;

}



3. Write a program to accept a character, an integer n and display the next n characters.

#include<iostream>

using namespace std;

int main(){

char ch;

int n,i=0;

cout<<"Enter Character : "<<endl;

cin>>ch;

cout<<"Enter N (n next characters) : "<<endl;

cin>>n;

// for(int i = 0; i!=n; i++){

// cout<<ch<" ";

// ch++;

// }

while(i!=n){

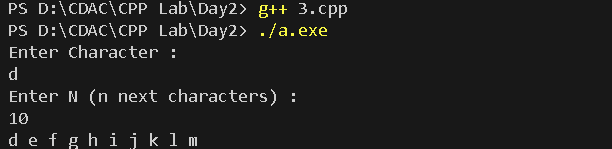
cout<<ch<<" ";

ch++; i++;

}

return 0;

}



4. Write a program to calculate factorial of a number.

For e.g. factorial of 5 = 5! = 5 \*4\*3\*2\*1 = 120.

#include<iostream>

using namespace std;

int main(){

int n, fact=1;

cout<<"Enter Number : "<<endl;

cin>>n;

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

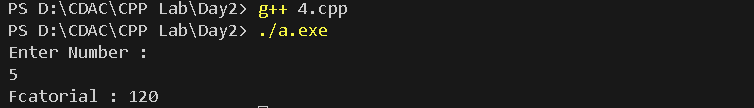
fact = fact \* i ;

}

cout<<"Fcatorial : "<<fact;

return 0;

}



5. Write a program to calculate factors of a given number.

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"Enter Number : ";

cin>>n;

cout<<"Factors of"<<n<<" = ";

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

if(n%i == 0){

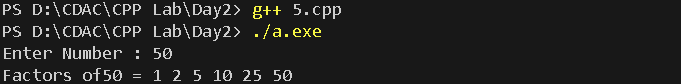
cout<<i<<" ";

}

}

return 0;

}



6. Accept two numbers and calculate GCD of them.

/\*6. Accept two numbers and calculate GCD of them.\*/

#include <bits/stdc++.h>

using namespace std;

int gcd(int a, int b)

{

int result = min(a, b);

while (result > 0) {

if (a % result == 0 && b % result == 0) {

break;

}

result--;

}

return result;

}

int main()

{

int a , b ;

cout<<"Enter A and B :"<<endl;

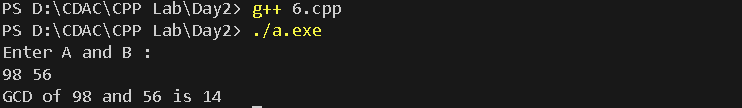
cin>>a>>b;

cout << "GCD of " << a << " and " << b << " is "

<< gcd(a, b);

return 0;

}



7. Write a menu driven program to do following operations :

a) Compute area of circle

b) Compute area of rectangle

c) Compute area of triangle

d) Exit

Display menu, ask choice to the user, depending on choice accept the parameters and perform the

operation. Continue this process until user selects exit option.

#include<iostream>

using namespace std;

class shape{

protected : double area\_res;

public:

virtual void area()=0;

};

class circle : public shape{

public: int r;

circle(){

r=1;

}

void area(){

area\_res = 3.14 \* r \*r;

cout<<"Area of Circle : "<<area\_res;

}

};

class Rectangle : public shape{

public: int l,b;

Rectangle(){

l = 1;

b = 1;

}

void area(){

area\_res = l\*b;

cout<<"Area of Rectangle : "<<area\_res;

}

};

class Triangle : public shape{

public: int b,h;

Triangle(){

h = 1;

b = 1;

}

void area(){

area\_res = 0.5\*b\*h;

cout<<"Area of Rectangle : "<<area\_res;

}

};

int main(){

shape \*sptr;

char ch;

// circle c\_obj;

// sptr = &c\_obj;

// sptr ->area();

circle c\_obj;

Rectangle r\_obj;

Triangle t\_obj;

do

{

cout<<"\n\n\_\_\_\_\_Menu\_\_\_\_\_\_ \na)Area of Circle \nb)Area of Rectangle \nc)Area of Triangle \nd)Exit \nEnter Choice : "<<endl;

cin>>ch;

switch (ch)

{

case 'a' :

cout<<"Enter radius : "<<endl;

cin>>c\_obj.r;

sptr = &c\_obj;

sptr -> area();

break;

case 'b':

cout<<"Enter Length and Breadth: "<<endl;

cin>>r\_obj.l>>r\_obj.b;

sptr = &r\_obj;

sptr -> area();

break;

case 'c' :

cout<<"Enter Base and Height: "<<endl;

cin>>t\_obj.b>>t\_obj.h;

sptr = &t\_obj;

sptr -> area();

break;

case 'd' :

cout<<"Exiting\_\_\_\_"<<endl; break;

default:

cout<<"Invalid choice"<<endl;

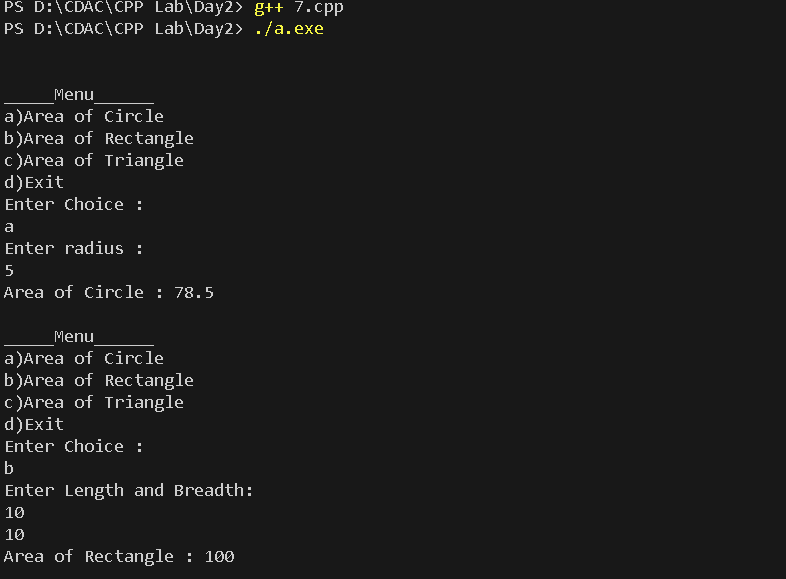
break;

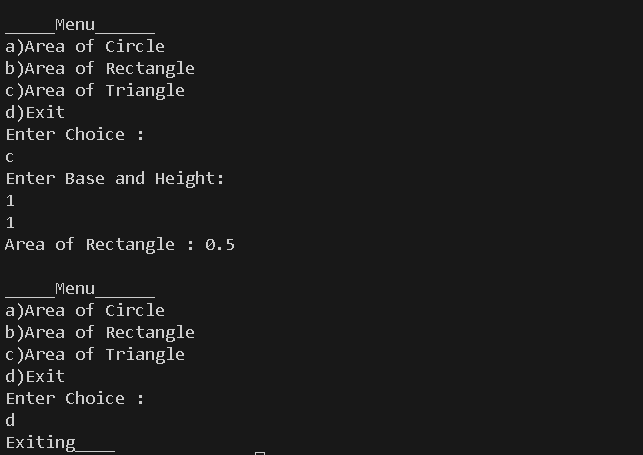
}

} while (ch!='d');

return 0;

}





8. Write a program to print all prime numbers between 1 to n

/\*8. Write a program to print all prime numbers between 1 to n\*/

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"Enter Value of N ;"<<endl;

cin>>n;

for(int i =2; i<=n; i++){

int flag =0;

for(int j=2; j<=i/2; ++j){

if(i%j ==0){

flag = 1;

break;

}

}

if(flag==0){

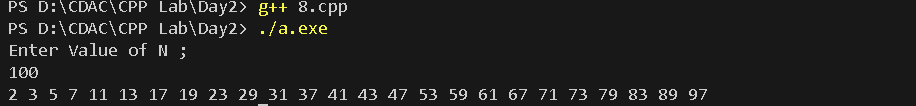
cout<<i<<" ";

}

}

return 0;

}



**Lab 3**

1:Write a program to create an array of integers and perform following operations on that array like

finding the sum, average, maximum and minimum number in that array. Accept the numbers of the

array from user.

#include<iostream>

using namespace std;

int main(){

int size, sum = 0;

cout<<"Enter Size of array\n";

cin>>size;

int arr[size];

cout<<"Enter array elements : "<<endl;

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

cout<<"Enter element at location : "<<i+1<<endl;

cin>>arr[i];

sum += arr[i];

}

double avg = sum/size;

cout<<"\nSum : "<<sum;

cout<<"\nAverage: "<<avg;

int min , max;

min = arr[0];

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

if(arr[i] < min){

min = arr[i];

}

}

max = arr[0];

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

if(arr[i] > max){

max = arr[i];

}

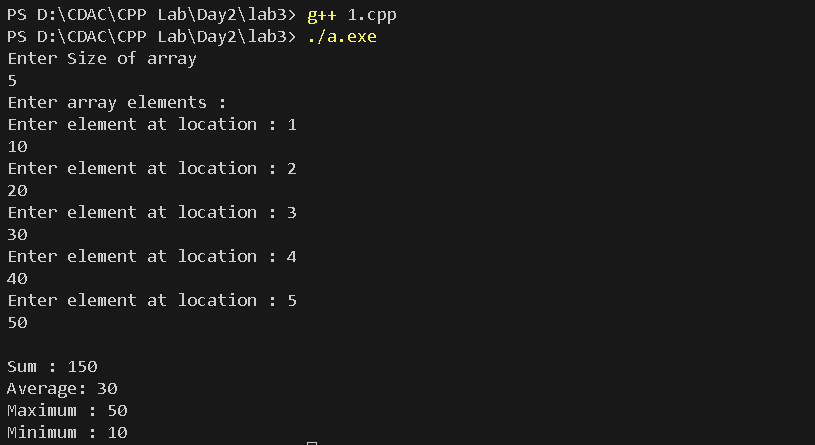
}

cout <<"\nMaximum : "<<max;

cout<<"\nMinimum : "<<min;

return 0;

}



2: Write a program to Accept a number and display its sum of digits.:ex 568 5+6+8

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter a number: ";

cin >> number;

int sum = 0;

int remainder;

while (number > 0) {

remainder = number % 10; // Extract the rightmost digit

sum += remainder; // Add the digit to the sum

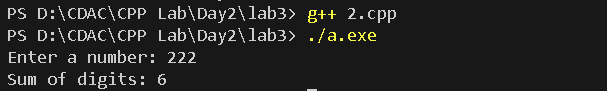
number /= 10; // Remove the rightmost digit

}

cout << "Sum of digits: " << sum << std::endl;

return 0;

}



3:. Write a program to find sum of all even and odd numbers between 1 to n.

/\*3:. Write a program to find sum of all even and odd numbers between 1 to n.\*/

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"Enter N :"<<endl;

cin>>n;

int even\_sum, odd\_sum;

even\_sum = odd\_sum = 0;

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

if(i%2 == 0){

even\_sum += i;

}

else{

odd\_sum += i;

}

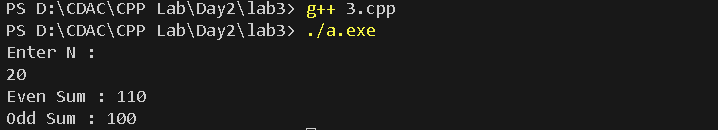
}

cout<<"Even Sum : "<<even\_sum<<endl;

cout<<"Odd Sum : "<<odd\_sum<<endl;

return 0;

}



4:. Write a program to print all Prime numbers between 1 to n.

#include<iostream>

using namespace std;

int main(){

int n;

cout<<"Enter N : "<<endl;

cin>>n;

for(int i =2; i<n ; i++){

int flag = 0;

for(int j =2; j<=i/2; ++j){

if(i%j == 0 ){

flag =1;

break;

}

}

if(flag==0){

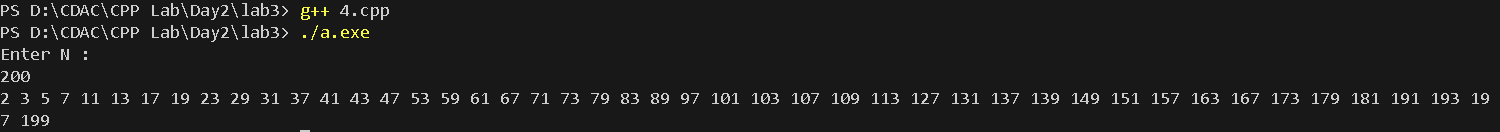
cout<<i<<" ";

}

}

return 0;

}



5:Write a program to accept array from user .Accept number from user and search number is present in array or not.

#include<iostream>

using namespace std;

int main(){

int arr[5], key;

cout<<"Enter Array Elemnts : "<<endl;

for(int i =0 ; i<5; i++){

cin>>arr[i];

}

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

cin>>key;

for(int i =0; i<5; i++){

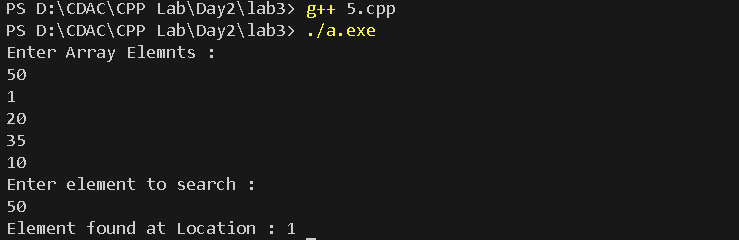
if(key == arr[i])

cout<<"Element found at Location : "<<i+1<<endl;

}

return 0;

}



6:Write a program to print following pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

#include <iostream>

int main() {

int rows;

std::cout << "Enter the number of rows: ";

std::cin >> rows;

// Outer loop for rows

for (int i = 1; i <= rows; ++i) {

// Inner loop for columns

for (int j = 1; j <= i; ++j) {

std::cout << "\* ";

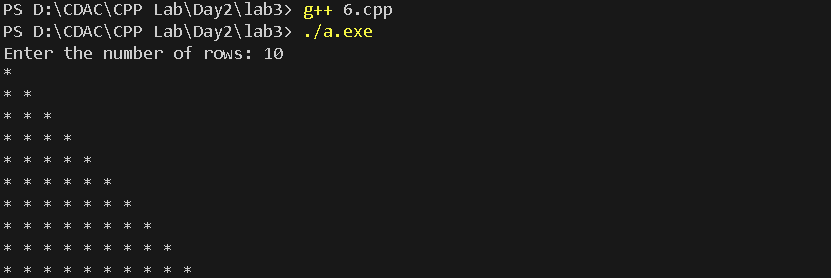
}

std::cout << std::endl; // Move to the next line after each row

}

return 0;

}



**Lab 4**

1:Write a program to create student class with data members rollno, marks1,mark2,mark3.

Accept data (acceptInfo()) and display using display member function.

Also display total,percentage and grade.

#include<iostream>

#include<string.h>

using namespace std;

class student{

    private:

        double rollNo,mark1,mark2,mark3,Total,AddTotal=300;

        double percentage;

        char grade = 'A';

        char \*name;

        void calcTotal(){

            Total = mark1 + mark2 + mark3;

        }

        void calcPercentage(){

            // percentage = 100;

            percentage = ((mark1+mark2+mark3)/300)\*100;

            // percentage = (mark)

        }

        void calcGrade(){

            if (percentage>80)

            {

                grade = 'A';

            }else if(percentage>60)

            {

                grade = 'B';

            }else{

                grade = 'C';

            }

        }

    public:

        student(){

            this->name=name;

            this->rollNo = rollNo;

            this->mark1 = mark1;

            this->mark2 = mark2;

            this->mark3 = mark3;

        }

        student(char \*name,int rollNo,int mark1,int mark2,int mark3){

            this->rollNo = rollNo;

            this->mark1 = mark1;

            this->mark2 = mark2;

            this->mark3 = mark3;

            this->name = name;

            this->name = new char[strlen(name)+1];

            strcpy(this->name,name);

        }

        void acceptInfo(char \*name,int rollNo,int mark1,int mark2,int mark3){

            this->rollNo = rollNo;

            this->mark1 = mark1;

            this->mark2 = mark2;

            this->mark3 = mark3;

        }

        void display(){

            calcTotal();

            calcPercentage();

            calcGrade();

            cout<<"Name: "<<name<<" RollNo: "<<rollNo<<" Mark1: "<<mark1<<" Mark2: "<<mark2<<" Mark3: "<<mark3<<" Total: "<<Total<<" Percentage: "<<percentage<<" Grade : "<<grade<<endl;

        }

        ~student(){

            cout<<"object destroyed"<<endl;

        }

        void setRollNo(int r){

            this->rollNo = r;

        }

        int getRollNo(){

            return this->rollNo;

        }

        void setName(char \*n){

            this->name = n;

        }

        string getName(char \*n){

            return this->name;

        }

        int getMark1(){

            return this->mark1;

        }

};

int main()

{

    // student s(101,80,50,70);

    int rollNo,mark1,mark2,mark3;

    char \*name;

    // cout<<"enter student details: ";

    // cin>>rollNo>>mark1>>mark2>>mark3;

    // student s(\*name,rollNo,mark1,mark2,mark3);

    // s.display();

    student dac[5];

    student s1("Kalyani",102,40,60,80);

   student s2("Bhakti",103,90,80,30);

student s3("Bhagawati",104,60,60,10);

    student s4("Hari",105,70,60,80);

    student s5("Maya",106,80,90,90);

    dac[0] = s1;

    dac[1] = s2;

    dac[2] = s3;

    dac[3] = s4;

    dac[4] = s5;

    cout<<"--------Display Student details-----"<<endl;

    for (int i = 0; i < 5; i++)

    {

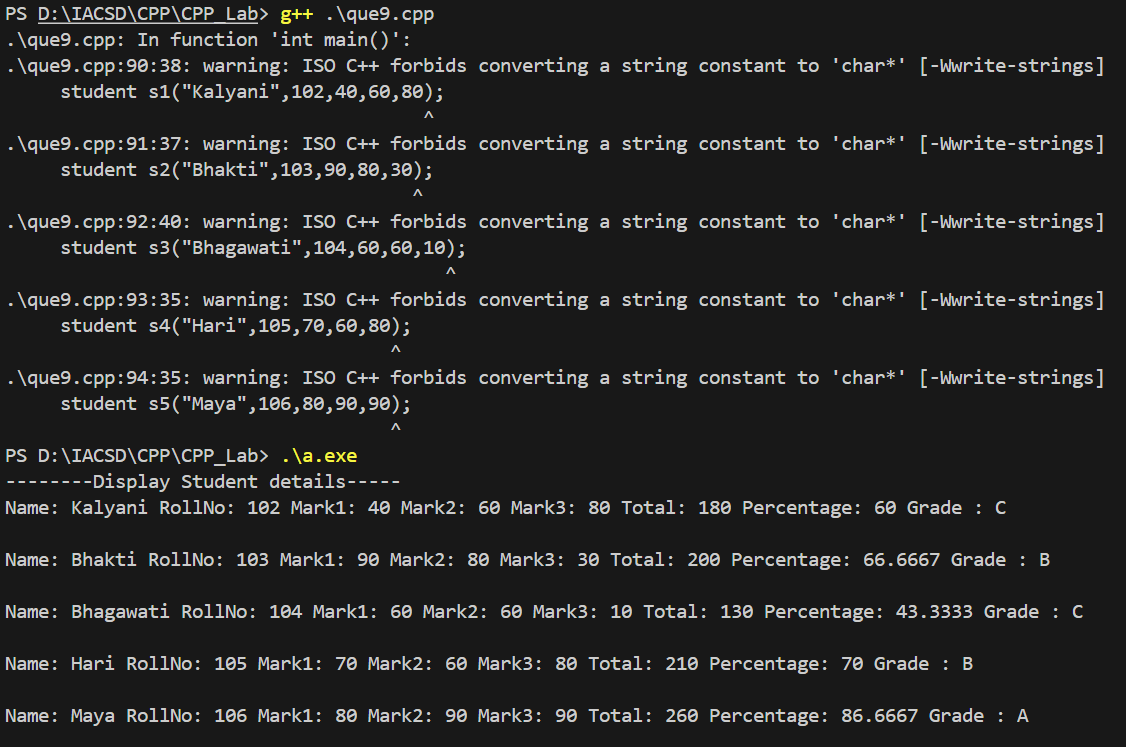
        dac[i].display();

        cout<<"\n";

    }

    return 0;

}



1. Create a class Person with data members as name, age, city. Write getters and setters for all the data

members. Also add the display function. Create Default and Parameterized constructors. Create the

object of this class in main method and invoke all the methods in that class.

#include<iostream>

#include<string.h>

using namespace std;

class person{

    int age;

    char \*name,\*city;

    public:

        person(){

            this->age=1;

            this->name="Ram";

            this->city = "Latur";

        }

        person(int age,char \*name,char \*city){

            this->age = age;

            this->name = new char[strlen(name)+1];

            strcpy(this->name,name);

            this->city = new char[strlen(city)+1];

            strcpy(this->city,city);

        }

        void display(){

            // for (int i = 0; i < strlen(name)+1; i++)

            // {

                // cout<<name;

            // }

            // for (int i = 0; i < strlen(city)+1; i++)

            // {

                // cout<<city;

            // }

        //    cout<<" Age: "<<age<<endl;

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

        }

void setAge(int age){

    this->age=age;

}

void setName(char \*name){

    this->name = name;

}

void setCity(char \*city){

    this->city = city;

}

int getAge(int age){

    return this->age;

}

string getName(char \*name){

    return this->name;

}

string getCity(char \*city){

    return this->city;

}

~person(){

    cout<<"object destryed"<<endl;

    delete name;

    delete city;

}

};

int main()

{

    person p;

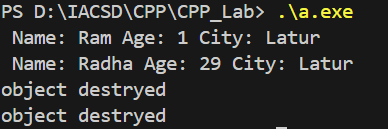
    p.display();

    person p1(29,"Radha","Latur");

    p1.display();

    return 0;

}



2. Create a class Date with data members as dd, mm, yy. Write getters and setters for all the data members. Also add the display function. Create Default and Parameterized constructors. Create the

object of this class in main method and invoke all the methods in that class.

#include<iostream>

using namespace std;

class date{

    private:

        int dd,mm,yy;

    public:

        date(){

            this->dd=1;

            this->mm=1;

            this->yy=2000;

        }

        date(int day,int month,int year){

            this->dd=day;

            this->mm=month;

            this->yy=year;

        }

        void display(){

            // cout<<"Day: "<<dd<<" Month:"<<mm<<" Year: "<<yy<<endl;

            cout<<dd<<"/"<<mm<<"/"<<yy<<endl;

        }

void setDay(int day){

    this->dd=day;

}

void setMonth(int month){

    this->mm=month;

}

void setYear(int year){

    this->yy=year;

}

int getDay(int day){

    return this->dd;

}

int getMonth(int month){

    return mm; //this pointer is provided by compiler itself

}

int getYear(int year){

    return this->yy;

}

~date(){

    cout<<"Objejt destroyed"<<endl;

}

};

int main()

{

    date d;

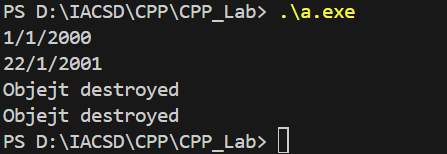
    d.display();

    date d1(22,1,2001);

    d1.display();

    return 0;

}



3. Create a class Book with data members as bname,id,author,price. Write getters and setters for all the

data members. Also add the display function. Create Default and Parameterized constructors. Create

the object of this class in main method and invoke all the methods in that class.

#include<iostream>

#include<string.h>

using namespace std;

class book{

    int id;

    float price;

    char \*bname,\*author;

    public:

        book(){

            this->bname="Atomic habit";

            this->author="James clare";

            this->id=0;

            this->price=0.1;

        }

        book(char \*bname,char \*author,int id,float price){

            this->bname=bname;

            bname = new char[strlen(bname)];

            strcpy(bname,bname);

            this->author=author;

            author = new char[strlen(author)];

            strcpy(author,author);

            this->id=id;

            this->price=price;

            // cout<<"Constrr created";

        }

        void display(){

            cout<<"Name: "<<bname<<" Author: "<<author<<" Id: "<<id<<" Price: "<<price<<endl;

        }

void setName(char \*bname){

    this->bname = bname;

}

void setAuthor(char \*author){

    this->author = author;

}

void setId(int id){

    this->id = id;

}

void setPrice(float price){

    this->price = price;

}

string getName(char \*name){

    return this->bname;

}

string getAuthor(char \*name){

    return this->author;

}

int getId(int id){

    return this->id;

}

float getPrice(int p){

    return this->price;

}

~book(){

    cout<<"object destroyed";

    delete bname;

    delete author;

}

};

int main()

{

    book b;

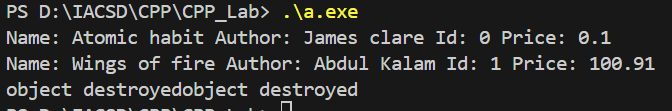
    b.display();

    book b1("Wings of fire","Abdul Kalam",1,100.91);

    b1.display();

    return 0;

}



4. Create a class Point with data members as x,y. Create Default and Parameterized constructors. Write

getters and setters for all the data members. Also add the display function. Create the object of this

class in main method and invoke all the methods in that class.

#include<iostream>

using namespace std;

// in this ex

//  this pointer is passed by compiler itself that's why not mentioned explicitly

class point{

    private:

        int x,y;

    public:

        point(){

            x=0;

            y=0;

        }

        point(int x1,int y1){

            x = x1;

            y = y1;

        }

        void display(){

            cout<<"X: "<<x<<" Y: "<<y<<endl;

        }

void setX(int x1){

    x = x1;

}

void setY(int y1){

    y = y1;

}

void getX(int x1){

    x = x1;

}

void getY(int y1){

    y = y1;

}

~point(){

    cout<<"object destroyed\n";

}

};

int main()

{

    int x,y;

    cout<<"enter point: ";

    cin>>x>>y;

    point p;

    p.display();

    point p1(1,8);

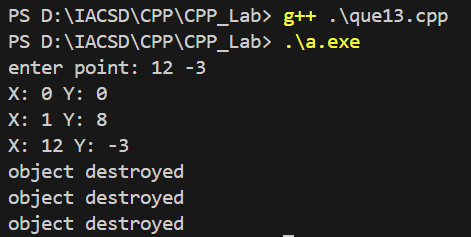
    p1.display();

    point p2(x,y);

    p2.display();

    return 0;

}



5. Create a class ComplexNumber with data members real, imaginary. Create Default and Parameterized constructors. Write getters and setters for all the data members. Also add the display function. Create the object of this class in main method and invoke all the methods in that class.

#include<iostream>

using namespace std;

class ComplexNumber{

    int real,imaginary;

    public:

        ComplexNumber(){

            this->real = 0;

            this->imaginary= 0;

        }

        ComplexNumber(int r,int imag){

            this->real = r;

            this->imaginary = imag;

        }

        void display(){

            // cout<<"Real: "<<real<<" Imaginary: "<<imaginary<<endl;

            cout<<"Complex no= "<<real<<"+i"<<imaginary<<endl;

        }

void setReal(int r){

    this->real = r;

}

void setImaginary(int i){

    this->imaginary = i;

}

int getReal(int r){

    return this->real;

}

int getImaginary(){

    return this->imaginary;

}

~ComplexNumber(){

    cout<<"Object destroyd\n";

}

};

int main()

{

    ComplexNumber c;

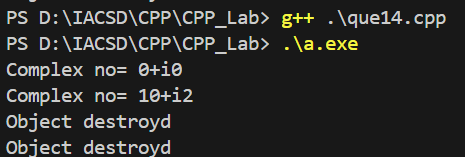
    c.display();

    ComplexNumber c1(10,2);

    c1.display();

    return 0;

}



**Lab 5**

Animal Hierarchy:

Problem Statement: Create a hierarchy of animal classes. Start with a base class Animal and then create derived classes like Mammal, Bird, and Fish. Each of these derived classes should have specific properties and methods related to their respective categories of animals.

#include<iostream>

using namespace std;

class animal{

    private:

        bool isBreath;

        string action;

        int age;

    public:

        animal(){

    this->isBreath = 0;

    action = "Eating";

    age = 1;

}

    animal(bool isBreath,string action,int age){

    this->isBreath = isBreath;

    this->action = action;

    this->age = age;

}

void display(){

    cout<<"IsBreath: "<<isBreath<<" Action: "<<action<<" Age: "<<age<<endl;

}

};

class mamal:public animal{

    string HairClr,colour;

    public:

        mamal(){

            cout<<"------default mamal---------\n";

            this->colour =" black";

            this->HairClr = "black";

        }

        mamal(bool isBreath,string action,int age,string HairClr,string colour):animal(isBreath, action, age){

            cout<<"-----parametarized mamal---------\n";

            this->colour = colour;

            this->HairClr = HairClr;

        }

        void displayMamal(){

            cout<<"Hair colour "<<HairClr<<" colour "<<colour<<endl;

        }

};

class bird:public animal{

    string HairClr,colour;

    public:

        bird(){

            cout<<"------default bird---------\n";

            this->colour = "white";

            this->HairClr = "white";

        }

        bird(bool isBreath,string action,int age,string HairClr,string colour):animal(isBreath, action, age){

            cout<<"-----parametarized bird---------\n";

            this->colour = colour;

            this->HairClr = HairClr;

        }

        void displaybird(){

            cout<<"Hair colour "<<HairClr<<" colour "<<colour<<endl;

        }

};

class fish:public animal{

    string colour;

    public:

        fish(){

            cout<<"------default fish---------\n";

            this->colour = "golden";

        }

        fish(bool isBreath,string action,int age,string colour):animal(isBreath, action, age){

            cout<<"-----parametarized fish---------\n";

            this->colour = colour;

        }

        void displayfish(){

            cout<<" colour "<<colour<<endl;

        }

};

int main(){

//  animal obj;

//  obj.display();

//  animal obj1(true,"walk",10);

//  obj1.display();

    cout<<"\n==========mamal===========\n";

    mamal obj2;

    obj2.display();

    obj2.displayMamal();

    mamal obj3(1,"dance",20,"black","white");

    obj3.display();

    obj3.displayMamal();

    cout<<"\n==========Bird===========\n";

    bird obj4(1,"fly",2,"blue","black");

    obj4.display();

    obj4.displaybird();

    cout<<"\n==========fish===========\n";

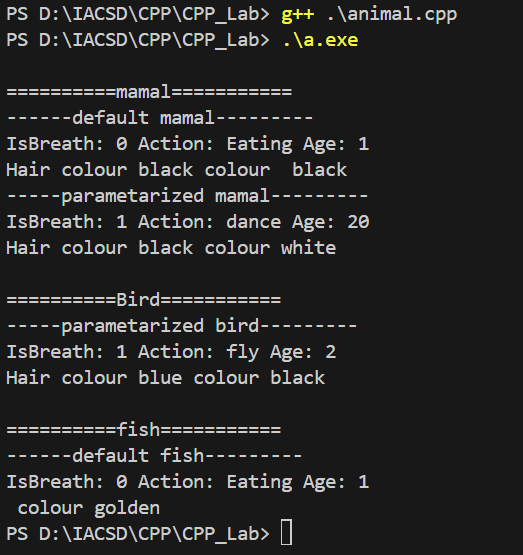
    fish obj5;

    obj5.display();

    obj5.displayfish();

    return 0;

}



Shape Hierarchy:

Problem Statement: Design a hierarchy of shape classes. Begin with a base class Shape and then create derived classes like Circle, Rectangle, and Triangle. Each shape should have methods for calculating area and perimeter specific to its geometry.

/\*Shape Sorting with Interfaces:

Problem Statement: Implement a shape sorting program. Define a base class Shape with properties like area and perimeter. Create derived classes like Circle, Rectangle, and Triangle. Implement an interface Sortable with a method to compare shapes by area. Use this interface to sort a list of shapes.

\*/

#include<iostream>

using namespace std;

class shape{

    int area,perimeter;

    public:

    virtual void display()=0;

    // virtual void calcArea()= 0;

    // virtual void calPerimeter()= 0;

};

class circle : public shape{

    int rad=10;

    public:

    void display(){

        cout<<"Area : "<<3.14\*rad\*rad<<endl;

        cout<<"Perimeter : "<<2\*3.14\*rad<<endl;

    }

};

class rectangle : public shape{

    int l =10, b=10;

    public:

    void display(){

        cout<<"Area : "<<l\*b<<endl;

        cout<<"Perimeter : "<<2\*(l+b)<<endl;

    }

};

class triangle : public shape{

    int height =12,breadth = 2,a=2,b=12,c=22;

    public:

    void display(){

        cout<<"Area : "<<(height\*breadth)/2<<endl;

        cout<<"Perimeter : "<<a+b+c<<endl;

    }

};

int main(){

    cout<<"======circle========\n";

    circle c;

    c.display();

    cout<<"=======rectangle======\n";

    rectangle r;

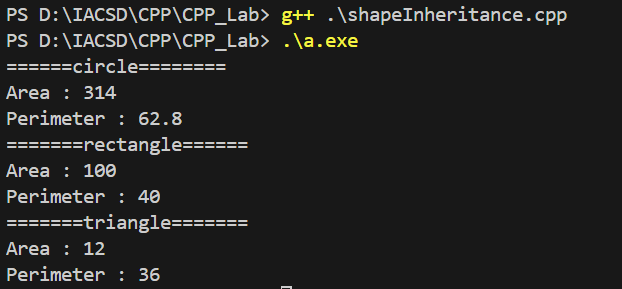
    r.display();

    cout<<"=======triangle=======\n";

    triangle t;

    t.display();

}



Employee Inheritance:

Problem Statement: Build a system for managing employees. Create a base class Employee with attributes such as name, employee ID, and salary. Then, derive classes like Manager and Developer, each with its own attributes and methods. Implement a common method, like calculate\_salary(), in the base class.

#include<iostream>

using namespace std;

class employee{

    int empId;

    string name;

    protected: double salary;

    public:

    employee(int id,string n,double s){

        empId = id;

        name = n;

        salary = s;

    }

        void calSalary(){

            cout<<"Salary: "<<salary<<endl;

        }

};

class developer:public employee{

    int deptNo;

    public:

        developer(int empId,string name,double salary,int deptNo):employee(empId,name,salary){

            this->deptNo = deptNo;

        }

        void developerTask(){

            cout<<"Developer specific task"<<endl;

        }

        // void calSalary(){

        //     employee ::calSalary();

        // }

};

class manager:public employee{

    int incentives;

    public:

        manager(int empId,string name,double salary,int incentives):employee(empId,name,salary){

            this->incentives = incentives;

        }

        void managerTask(){

            cout<<endl<<"Manager specific task"<<endl;

        }

        void calSalary(){

            // employee::calSalary();

            cout<<"Salary of manager: "<<this->salary+this->incentives;

        }

};

int main()

{

    // employee e(101,"Kalyani Kadam",100000);

    // e.calSalary();

    cout<<"\_\_\_\_\_\_\_\_\_Developer\_\_\_\_\_\_\_\_\_\_"<<endl;

    developer d(102,"Kadam Hari",90000,4);

    d.calSalary();

    d.developerTask();

    cout<<"\_\_\_\_\_\_\_\_\_Manager\_\_\_\_\_\_\_\_\_\_\_"<<endl;

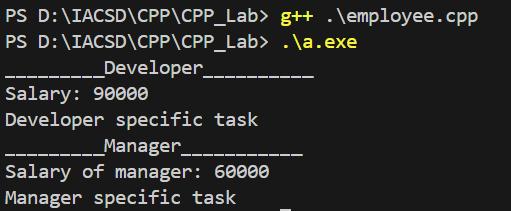
    manager m(10,"Patil Sam",50000,10000);

    m.calSalary();

    m.managerTask();

    return 0;

}



Vehicle Inheritance:

Problem Statement: Develop a class hierarchy for vehicles. Start with a base class Vehicle and create derived classes like Car, Motorcycle, and Truck. Each derived class should have unique properties like the number of wheels and specific methods like start\_engine().

#include<iostream>

using namespace std;

class vehicle{

    int wheel;

    string fuelType,name;

    public:

        vehicle(){

            wheel = 0;

            fuelType = "no fuel";

            name = "no name";

        }

        vehicle(int w,string ft,string n){

            this->wheel = w;

            this->fuelType = ft;

            this->name = n;

        }

        void display(){

            cout<<" Wheel "<<wheel<<" Fuel Type: "<<fuelType<<" Name :"<<name<<endl;

        }

};

class car : public vehicle{

    string sensor;

    public:

        car(){

            sensor = " no sensor";

        }

        car(int w,string ft,string n,string sensor):vehicle(w,ft,n){

            this->sensor = sensor;

        }

        void display(){

            vehicle::display();

            cout<<" Sensor: "<<sensor<<endl;

        }

};

class truck: public vehicle{

    int loadCapacity;

    public:

        truck(){

            loadCapacity = 0;

        }

        truck(int w,string ft,string n,int loadCapacity):vehicle(w,ft,n){

            this->loadCapacity = loadCapacity;

        }

        void display(){

            vehicle::display();

            cout<<" loadCapacity: "<<loadCapacity<<endl;

        }

};

int main(){

//  vehicle v;

//  v.display();

//  car c(4,"diesel","bmw","temp");

//  c.display();

int w,lc;

static int i=0;

string ft,n,s;

vehicle \*veh[2];

car\* c;

truck\* t;

    cout<<"\n0.exit\n1.add car details\n2.add truck details: "<<endl;

    int ch;

    do{

        cout<<"Enter choice: ";

        cin>>ch;

    switch(ch){

    case 1:

    if(i<2){

        cout<<"Enter car details: wheel fuelType name sensor---"<<endl;

        cin>>w>>ft>>n>>s;

        c =new car(w,ft,n,s);

        veh[i] = c;

        veh[i]->display();

        i++;

//      veh[i]->

    }

    else{

        cout<<"Capacity full";

    }

    break;

    case 2:

        if(i<2){

        cout<<"Enter truck details: wheel fuelType name loadCapacity---"<<endl;

        cin>>w>>ft>>n>>lc;

         t =new truck(w,ft,n,lc);

        veh[i] = t;

        veh[i]->display();

        i++;

//      veh[i]->

    }

    else{

        cout<<"Capacity full";

    }

    break;

    default:

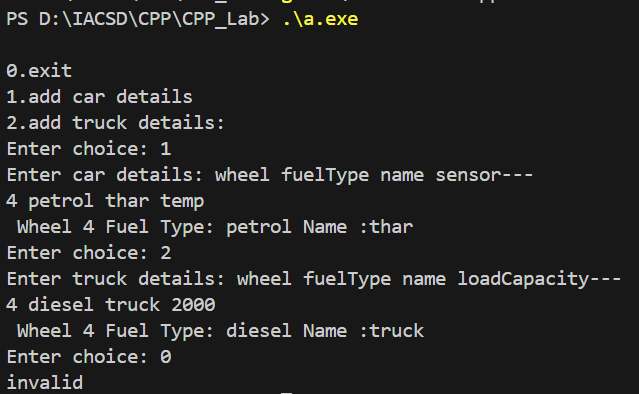
        cout<<"invalid";

        break;

    }

    }while(ch!=0);

    }



Bank Account Inheritance:

Problem Statement: Design a system for managing bank accounts. Create a base class BankAccount with attributes like account number and balance. Derive classes like SavingsAccount and CheckingAccount, each with specialized methods like withdraw() and calculate\_interest().

#include<iostream>

using namespace std;

class bankAccount{

    int accNo;

    protected: double balance;

    public:

        bankAccount(int accNo,double balance){

            this->accNo = accNo;

            this->balance = balance;

        }

        void display(){

            cout<<"Account no: "<<accNo<<" Balance: "<<balance<<endl;

        }

};

class savingAccount:public bankAccount{

    int interest,rate,time;

    public:

        savingAccount(int accNo,double balance,int rate,int time):bankAccount(accNo,balance){

            this->rate = rate;

            this->time = time;

        }

        void calInterest(){

            // cout<<"Enter rate of interest & time period: ";

            // cin>>rate>>time;

            interest = (balance\*rate\*time)/100;

            cout<<"Interest "<<interest<<endl;

        }

        void withdraw(int amt){

            balance = balance - amt;

            // cout<<"Balance: "<<balance<<endl;

        }

        void display(){

            bankAccount::display();

            cout<<"Interest "<<interest<<endl;

            cout<<"Balance: "<<balance<<endl;

        }

};

class checkingAccount:public bankAccount{

    int interest,rate,time;

    public:

        checkingAccount(int accNo,double balance,int rate,int time):bankAccount(accNo,balance){

            this->rate = rate;

            this->time = time;

        }

        void calInterest(){

            // cout<<"Enter rate of interest & time period: ";

            // cin>>rate>>time;

            interest = (balance\*rate\*time)/100;

            cout<<"Interest "<<interest<<endl;

        }

        void withdraw(int amt){

            balance = balance - amt;

            // cout<<"Balance: "<<balance<<endl;

        }

        void display(){

            bankAccount::display();

            cout<<"Interest "<<interest<<endl;

            cout<<"Balance: "<<balance<<endl;

        }

};

int main()

{

    cout<<"\_\_\_\_\_\_\_\_\_\_\_Saving account\_\_\_\_\_\_\_\_"<<endl;

    savingAccount sa(1010,8000,2,4);

    // sa.display();

    sa.calInterest();

    sa.withdraw(1000);

    sa.display();

    cout<<"\_\_\_\_\_\_\_\_\_\_\_Checking account\_\_\_\_\_\_\_\_"<<endl;

    checkingAccount ca(1902,28900,2.4,4);

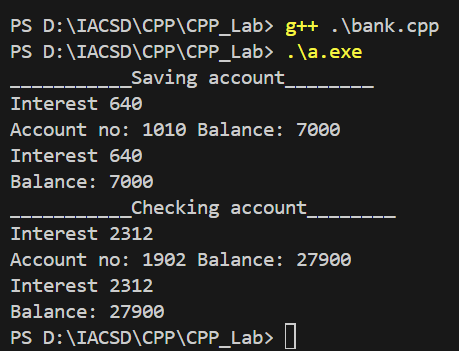
    ca.calInterest();

    ca.withdraw(1000);

    ca.display();

    return 0;

}



Geometric Shapes with Polymorphism:

Problem Statement: Extend the shape hierarchy example by implementing polymorphism. Define a base class Shape with methods to calculate area and perimeter. Then, create derived classes like Circle, Rectangle, and Triangle, each with its own implementation of these methods.

/\*Geometric Shapes with Polymorphism:

Problem Statement: Extend the shape hierarchy example by implementing polymorphism. Define a base class Shape with methods to calculate area and perimeter. Then, create derived classes like Circle, Rectangle, and Triangle, each with its own implementation of these methods\*/

#include<iostream>

using namespace std;

class shape{

    int area,perimeter;

    public:

        void calArea(){

            cout<<"Area: "<<area<<" perimeter: "<<perimeter<<endl;

        }

};

class circle:public shape{

    double radius;

    public:

        circle(int radius){

            this->radius = radius;

        }

        void calArea(){

            cout<<"Area of Circle: "<<3.14\*radius\*radius;

        }

};

int main()

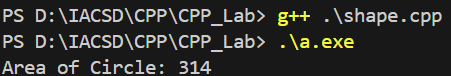
{

    circle c(10);

    c.calArea();

    return 0;

}



Person and Student Inheritance:

Problem Statement: Model a system for handling individuals and students within an educational institution. Create a base class Person with attributes like name and age. Derive a Student class with additional attributes like student ID and GPA, inheriting the common attributes from the Person class.

/\*Person and Student Inheritance:

Problem Statement: Model a system for handling individuals and students within an educational institution. Create a base

 class Person with attributes like name and age. Derive a Student class with additional attributes like student ID and GPA,

 inheriting the common attributes from the Person class.\*/

#include<iostream>

using namespace std;

class person{

    int age;

    string name;

    public:

        virtual void display(){

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

        }

        person(int a,string n){

            age = a;

            name = n;

        }

};

class student:public person{

    int studId;

    double gpa;

    public:

        student(int a,string n,int id,double gpa):person(a,n){

            studId = id;

            this->gpa = gpa;

        }

        void display(){

            person::display();

            cout<<" Stud id : "<<studId<<" GPA :"<<gpa<<endl;

        }

};

int main(){

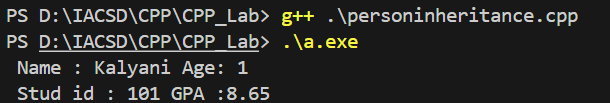
//  person p(1,"Kalyani");

//  p.display();

     student s(1,"Kalyani",101,8.65);

     s.display();

}



Library Catalog with Books and Journals:

Problem Statement: Build a library catalog system. Create a base class LibraryItem with properties like title and author. Then, derive classes like Book and Journal, each with their unique properties. Implement methods to check out and return items in the derived classes.

/\*Library Catalog with Books and Journals:

Problem Statement: Build a library catalog system. Create a base class LibraryItem with properties like title and author. Then, derive classes like Book and Journal, each with their unique properties. Implement methods to check out and return items in the derived classes.\*/

#include<iostream>

using namespace std;

class libraryItem

{

    // public:static int counter=0;

private:

    string title;

    string author;

public:

    libraryItem(string title,string author){

        this->title = title;

        this->author=author;

        // counter++;

    }

    virtual void display(){

        cout<<"Title :"<<title<<" author: "<<author<<endl;

    }

    ~libraryItem(){

        cout<<"destructor is called!"<<endl;

    }

};

class book:public libraryItem{

    int id;

    double price;

    public:

        book(string title,string author,int id,double price):libraryItem(title,author){

            this->id = id;

            this->price = price;

            // counter++;

        }

        void display(){

            libraryItem::display();

            cout<<" BOOK Id :"<<id<<" price : "<<price<<endl;

        }

};

class journals : public libraryItem{

    int JId;

    double price;

    public:

        journals(string title,string author,int id,double price):libraryItem(title,author){

            this->JId = id;

            this->price = price;

            // counter++;

        }

        void display(){

            libraryItem::display();

            cout<<" BOOK Id :"<<JId<<" price : "<<price<<endl;

        }

};

int main()

{

    book b1("java","james gosling",101,800.23);

    b1.display();

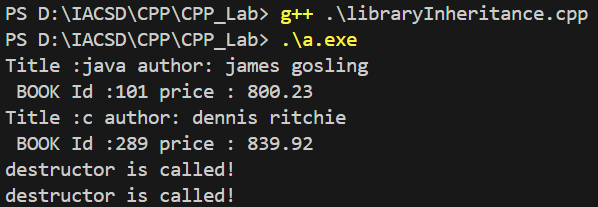
    journals j1("c","dennis ritchie",289,839.92);

    j1.display();

    // cout<<"count : "<<j1.counter<<endl;

    return 0;

}



Shape Sorting with Interfaces:

Problem Statement: Implement a shape sorting program. Define a base class Shape with properties like area and perimeter. Create derived classes like Circle, Rectangle, and Triangle. Implement an interface Sortable with a method to compare shapes by area. Use this interface to sort a list of shapes.

/\*Shape Sorting with Interfaces:

Problem Statement: Implement a shape sorting program. Define a base class Shape with properties like area and perimeter. Create derived classes like Circle, Rectangle, and Triangle. Implement an interface Sortable with a method to compare shapes by area. Use this interface to sort a list of shapes.

\*/

#include<iostream>

using namespace std;

class shape{

    int area,perimeter;

    public:

    virtual void display()=0;

    // virtual void calcArea()= 0;

    // virtual void calPerimeter()= 0;

};

class circle : public shape{

    int rad=10;

    public:

    void display(){

        cout<<"Area : "<<3.14\*rad\*rad<<endl;

        cout<<"Perimeter : "<<2\*3.14\*rad<<endl;

    }

};

class rectangle : public shape{

    int l =10, b=10;

    public:

    void display(){

        cout<<"Area : "<<l\*b<<endl;

        cout<<"Perimeter : "<<2\*(l+b)<<endl;

    }

};

class triangle : public shape{

    int height =12,breadth = 2,a=2,b=12,c=22;

    public:

    void display(){

        cout<<"Area : "<<(height\*breadth)/2<<endl;

        cout<<"Perimeter : "<<a+b+c<<endl;

    }

};

int main(){

    cout<<"======circle========\n";

    circle c;

    c.display();

    cout<<"=======rectangle======\n";

    rectangle r;

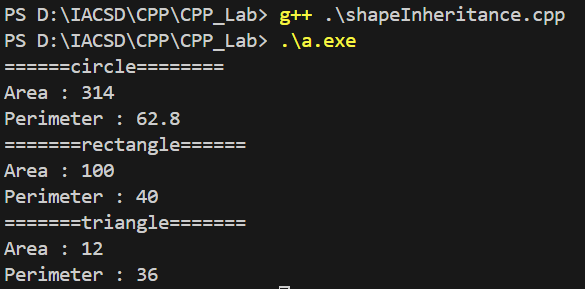
    r.display();

    cout<<"=======triangle=======\n";

    triangle t;

    t.display();

}



Employee Payroll System with Abstract Classes:

Problem Statement: Design an employee payroll system. Create an abstract class Employee with attributes like name and employee ID. Derive concrete classes like HourlyEmployee and SalariedEmployee. Define abstract methods for calculating pay in the base class and implement them in the derived classes.

/\*Employee Payroll System with Abstract Classes:

Problem Statement: Design an employee payroll system. Create an abstract class Employee with attributes like name and employee ID. Derive concrete classes like HourlyEmployee and SalariedEmployee. Define abstract methods for calculating pay in the base class and implement them in the derived classes.\*/

#include<iostream>

using namespace std;

class emp{

    int empId;

    string name;

    public:

        emp(int id,string n){

            empId = id;

            name = n;

        }

        virtual void calcPay()=0;

        void display(){

            cout<<"Emp id : "<<empId<<" Name: "<<name<<endl;

        }

};

class hourlyEmp : public emp{

    int rate,hrworked;

    public:

        hourlyEmp(int id,string name,int rate,int hrworked):emp(id,name){

            this->rate = rate;

            this->hrworked = hrworked;

        }

        void calcPay(){

            cout<<"Hourly employee : "<<rate\*hrworked<<endl;

        }

};

class salariedEmp: public emp{

    int salary;

    public:

        salariedEmp(int id,string name,int salary):emp(id,name){

            this->salary = salary;

        }

        void calcPay(){

            cout<<"Salary: "<<salary<<endl;

        }

};

int main()

{

    cout<<"\n=====Hourly employee======\n";

    hourlyEmp he(12,"Kalyani",10,10);

    he.display();

    he.calcPay();

    cout<<"\n=====salaried Employee=====\n";

    salariedEmp se(101,"Maya",100000);

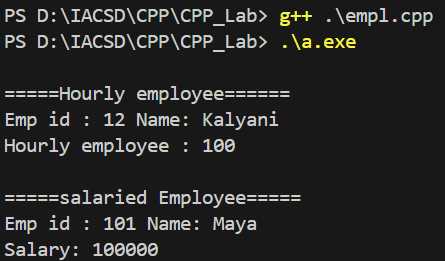
    se.display();

    se.calcPay();

    return 0;

}

\



These examples illustrate how inheritance can be used to model real-world hierarchies and relationships in object-oriented programming, promoting code reusability and organization. They also highlight various concepts like polymorphism, interfaces, and abstract classes to solve specific programming challenges.

**Lab 7:**

1 Solve this.

Fresh business scenario to apply inheritance , polymorphism to emp based organization scenario.

Create Emp based organization structure --- Emp , Mgr , Worker

1.1 Emp state--- id(int), name, deptId , basicSalary(double)

Accept all of above in constructor arguments.

Methods ---

1.2. compute net salary ---ret 0

(eg : public double computeNetSalary(){return 0;})

1.2 Mgr state ---id,name,basic,deptId , perfBonus

Add suitable constructor

Methods ----

1. compute net salary (formula: basic+perfBonus) -- override computeNetSalary

1.3 Worker state --id,name,basic,deptId,hoursWorked,hourlyRate

Methods :

1. compute net salary (formula: = basic+(hoursWorked\*hourlyRate) --override computeNetSalary

2. get hrlyRate of the worker -- add a new method to return hourly rate of a worker.(getter)

Create suitable array to store organization details.

Provide following options

1. Hire Manager

I/P : all manager details

2. Hire Worker

I/P : all worker details

3. Display information of all employees net salary (by invoking computeNetSal),

4. Exit

----------------------------------------------------

#include<iostream>

using namespace std;

class employee{

    private: int id,DeptId;string name;

    protected: double basicSalary;

    public:

        employee(){

            id = 0;

            name ="no name";

            DeptId = 0;

            basicSalary = 15000;

        }

        employee(int employeeId,int deptId,string name,double basicSalary){

            this->id = employeeId;

            this->name = name;

            this->DeptId = deptId;

            this->basicSalary = basicSalary;

        }

        virtual void computeNetSalary(){

            cout<<" Net salary of employee: "<<basicSalary<<endl;

            // return 0;

        }

        virtual void display(){

            cout<<" Id : "<<id<<" Name : "<<name<<" DeptId : "<<DeptId;

        }

        void setId(int id){

            this->id = id;

        }

        void setName(string name){

            this->name = name;

        }

        void setDeptId(int deptId){

            this->DeptId = deptId;

        }

        void setSalary(double salary){

            this->basicSalary = salary;

        }

        int getId(){

            return this->id;

        }

        int getdeptId(){

            return this->DeptId;

        }

        string getName(){

            return this->name;

        }

        double getSalary(){

            return this->basicSalary;

        }

};

class manager : public employee{

    double perfBonus;

    public:

        manager(){

            perfBonus = 0;

        }

        manager(int id,int deptId,string name,double perfBonus,double basicSalary):employee(id,deptId,name,basicSalary){

            this->perfBonus = perfBonus;

        }

        void computeNetSalary(){

            // employee::computeNetSalary();

            double netSalary = basicSalary+perfBonus;

            cout<<" Net salary of manager: "<<netSalary;

            // return 0;

        }

        void display(){

            // cout<<" perfBonus : "<<perfBonus<<endl;

            employee::display();

            cout<<" perfBonus : "<<perfBonus<<endl;

        }

        void setPerfBonus(double perfBonus){

            this->perfBonus = perfBonus;

        }

        double getPerfBonus(){

            return this->perfBonus;

        }

};

class worker:public employee{

    int hoursWorked, hourlyRate;

    public:

    worker(){

        hourlyRate = 0;

        hoursWorked = 0;

    }

     worker(int id,int deptId,string name,double basicSalary,int hourlyRate,int hoursWorked):employee(id,deptId,name,basicSalary){

        this->hourlyRate = hourlyRate;

        this->hoursWorked = hoursWorked;

    }

    void setHourlyRate(int hourlyRate){

        this->hourlyRate = hourlyRate;

    }

    int getHourlyRate(){

        return this->hourlyRate;

    }

    void computeNetSalary(){

        double netSalary = basicSalary+(hoursWorked\*hourlyRate);

        cout<<" Net salary of worker : "<<basicSalary+hoursWorked\*hourlyRate<<endl;

        // cout<<" Net salary of worker : "<<netSalary<<endl;

        // return 0;

    }

    void display(){

        employee::display();

        cout<<" Hourly rate of worker: "<<this->hourlyRate<<" worked hours: "<<this->hoursWorked<<endl;

    }

    void setHoursWorked(int hourlyRate){

        this->hoursWorked = hoursWorked;

    }

    int getHoursWorked(){

        return this->hoursWorked;

    }

};

int main(){

    // employee e;

    // e.computeNetSalary();

    manager m;

    // m.computeNetSalary();

    // cout<<"\n--------manager-----"<<endl;

    // manager m1(101,2,"Kalyani",10000,25000);

    // m1.display();

    // m1.computeNetSalary();

    // cout<<"\n--------worker-----"<<endl;

    worker w;

    // w.display();

    // w.computeNetSalary();

    // worker w1(5,"Kalyani",3,4,13,50000);

    // w1.display();

    // w1.computeNetSalary();

    // w1.getHourlyRateOfWorker(12);

    employee \*emp[2];

    emp[1] =  &m;

    emp[2] = &w;

    // emp[3] = &w;

    int id,DeptId,hourlyRate,hoursWorked;

    string name;

    double basicSalary;

    double perfBonus;

    int ch,count;;

    do

    {

        cout<<"\n1.Hire manager\n2.Hire worker\n3.Display information of all employees net salary\n4.Exit\n";

        cout<<endl<<"Enter choice: ";

        cin>>ch;

        // for(int i=1;i<5;i++)

        // {

            switch(ch){

            case 1:{

                // int count;

                cout<<"How many manager details do you want to add: ";

                cin>>count;

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

                {

                    cout<<"Enter manager details: id>>DeptId>>name>>perfBonus>>basicSalary: "<<endl;

                    cin>>id>>DeptId>>name>>perfBonus>>basicSalary;

                }

                manager mgr(id,DeptId,name,perfBonus,basicSalary);

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

                {

                    emp[i] = &mgr;

                    emp[i]->display();

                    emp[i]->computeNetSalary();

                    cout<<endl;

                }

                cout<<"\n-------------------\n"<<endl;

                break;

            }

            case 2:

                {

                cout<<"How many worker details do you want to add: id>>DeptId>>name>>basicSalary>>hourlyRate>>hoursWorked: ";

                cin>>count;

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

                {

                    cout<<"Enter worker details: "<<endl;

                    cin>>id>>DeptId>>name>>basicSalary>>hourlyRate>>hoursWorked;

                }

                worker wkr(id,DeptId,name,basicSalary,hourlyRate,hoursWorked);

                // worker wkr(id,name,DeptId,hourlyRate,hoursWorked,basicSalary);

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

                {

                    emp[i] = &wkr;

                    emp[i]->display();

                    emp[i]->computeNetSalary();

                }

                cout<<"\n-------------------\n"<<endl;

                break;

                }

            case 3:

                {

                    cout<<" enter salary manager: ";

                    cin>>basicSalary;

                    manager m(id,DeptId,name,perfBonus,basicSalary);

                    emp[1]=&m;

                    cout<<" enter salary worker: ";

                    cin>>basicSalary;

                    worker w(id,DeptId,name,basicSalary,hourlyRate,hoursWorked);

                    emp[2]=&w;

                    for (int i = 1; i < 3; i++)

                    {

                        // emp[i] = &wkr;

                    // emp[i]->display();

                    // emp[i]->computeNetSalary();

                    emp[i]->computeNetSalary();

                    }

                   break;

                }

            case 4:

                exit(0);

                break;

            default:

                cout<<"Invalid";

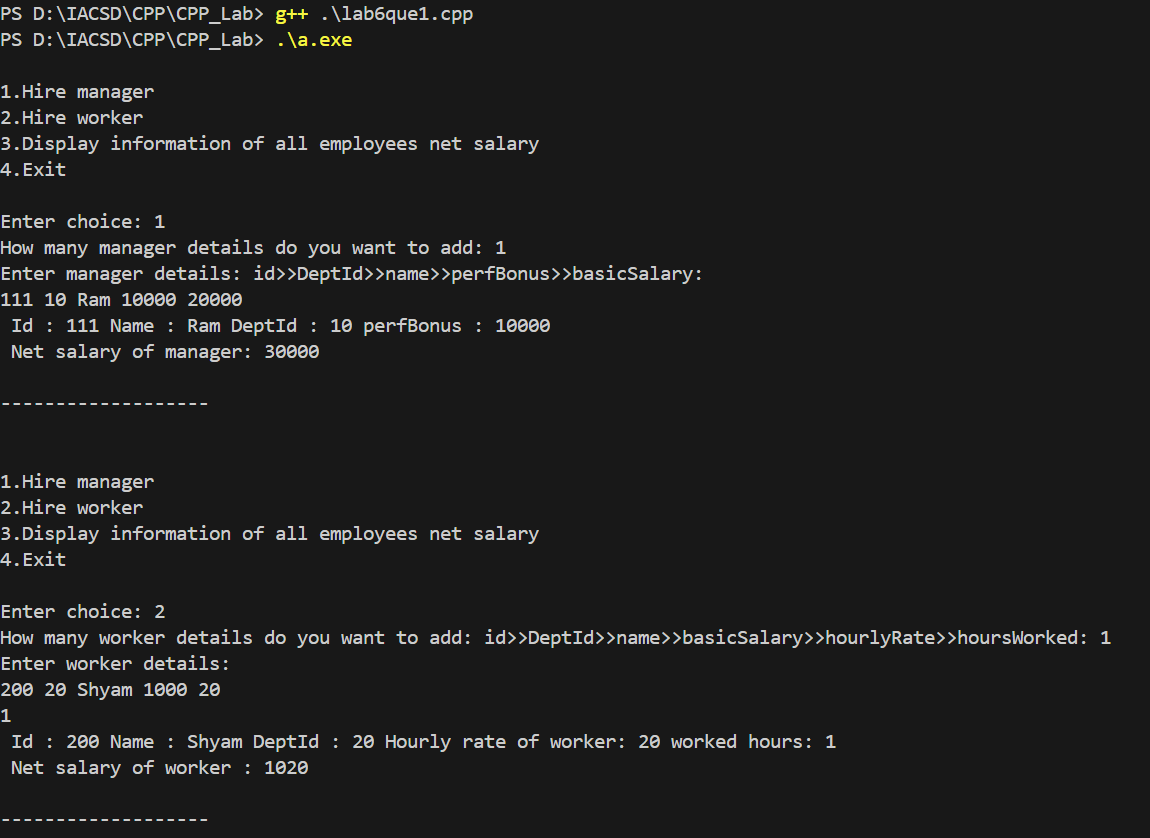
                break;

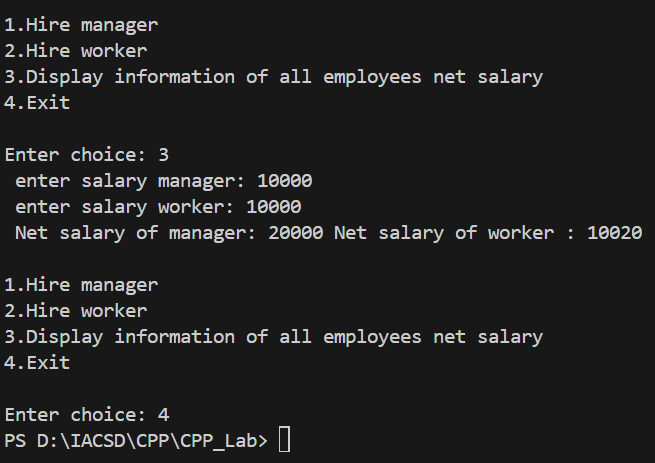
        }

        // }

    } while (ch!=4);

}





2:Create cpp application for bank account handling.

2.1. Create a class BankAccount -- acct no(int),customer name(string),balance(double)

Add constr. (2 constrs : first to accept all details )

2.2 Add Business logic methods

Methods

public void withdraw(double amt)

public void deposit(double amt)

2.3: Create object of account class and test withdraw and deposit methods.

---------------------------------------------------------------------------------

#include<iostream>

using namespace std;

class bankAccount{

    int accNo;

    string name;

    double balance;

    public:

        bankAccount(){

            accNo = 0;

            name = "no name";

            balance = balance;

        }

        bankAccount(int no,string nam,double bal){

            accNo = no;

            name = nam;

            balance = bal;

        }

        void withdraw(double amount){

            balance = balance - amount;

        }

        void deposit(double amount){

            balance = balance + amount;

        }

        void display(){

            cout<<" Balance :"<<balance<<endl;

        }

};

class account:public bankAccount{

    int amt;

    public:

    account(int n,string naam,double bal,double amt):bankAccount(n,naam,bal){

            this->amt= amt;

    }

};

int main()

{

    // bankAccount bac;

    // bac.withdraw(1000);

    // bac.display();

    // account b(101,"Kalyani",10000,2000);

    account b(10,"KKM",20000,1000);

    b.deposit(1000);

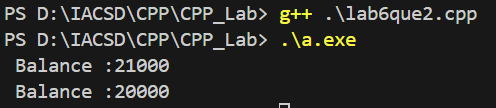
    b.display();

    b.withdraw(1000);

    b.display();

    return 0;

}



3:Create a abstract class Shape with pure virtual method area;

Create Rectangle,Circle,Square class..inherit them from Shape class..Override area method.

Test these all classes by creating object of respective class.

#include<iostream>

using namespace std;

class shape{

    public:

        virtual void area() = 0;

        // area();

        void print(){

            cout<<"\n\*\*\*\*\*\*\*"<<endl;

        }

};

class rectangle:public shape{

    double length,breadth;

    public :

        rectangle(double length, double breadth){

            this->length = length;

            this->breadth = breadth;

        }

        void area(){

            cout<<"Area of rectangle: "<<length\*breadth<<endl;

        }

};

class circle : public shape{

    double radius,pi=3.14;

    public:

        circle(double rad){

            radius = rad;

        }

        void area(){

            cout<<"Area of circle: "<<pi\*radius\*radius<<endl;

        }

};

class triangle : public shape{

    double height,width;

    public:

        triangle(double height,double width){

            this->height = height;

            this->width = width;

        }

        void area(){

            cout<<"Area of triangle: "<<(height\*width)/2;

        }

};

int main()

{

    // shape s;

    rectangle r(10,20);

    r.area();

    r.print();

    circle c(10);

    c.area();

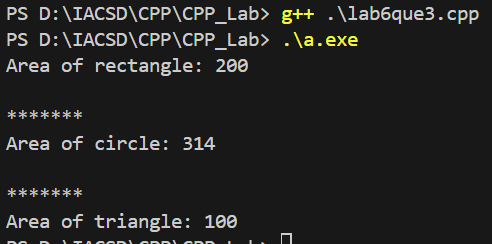
    c.print();

    triangle t(10,20);

    t.area();

    return 0;

}



**Lab 7**

assignment:Exception

create Stack class for storing 10 numbers

create function push(int number)--->number will get stored in array

create function pop() will return top most number ,last added number

Note:Hadle StackFull and StackEmpty Exception

Stack s;//array[4];

s.push(10);s.push(20);s.push(30);//s.push(40); s.push(50);

int a=s.pop();//------> 30

a=s.pop();//----->20

a=s.pop();//------>10

s.pop();//

#include<iostream>

using namespace std;

class stack{

    // int s;

    // int arr[s];

    public:

        int size,top = -1;

        int \*arr = new int(size);

        void setSize(int size){

            this->size = size;

        }

        void push(int value){

            if(isFull()){

                cout<<"Stack is full"<<endl;

            }else{

                top++;

                // cout<<"Enter Element : "<<endl;

                // cin>>arr[top];

                arr[top] = value;

                cout<<"Element added!"<<endl;

            }

        }

        void pop(){

            if (isEmpty())

            {

                cout<<"Stack is empty!"<<endl;

            }else{

                cout<<arr[top]<<" removed"<<endl;

                arr[top] = 0;

                top--;

            }

        }

        int isFull(){

            if (top == size-1)

            {

                return 1;

            }else{

                return 0;

            }

        }

        int isEmpty(){

            if(top==-1){

                return 1;

            }else{

                return 0;

            }

        }

        // void add(){

        //     cout<<"Enter "<<sizeof(arr)/sizeof(int)<<" element: ";

        //     for (int i = 0; i < sizeof(arr)/sizeof(int); i++)

        //     {

        //         cin>>arr[i];

        //     }

        // }

        void display(){

            cout<<endl<<"-----------Stack elements----------"<<endl;

            if (isEmpty())

                {

                    cout<<"Stack is empty"<<endl;

                }else{

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

                    {

                        cout<<arr[i]<<" ";

                    }

                }

        }

};

int main()

{

    stack s1;

    // s1.pop();

    int size,ch,element;

    cout<<"Enter size: ";

    cin>>size;

    s1.setSize(size);

    // s1.push(89);

    // s1.push(67);

    // s1.push(63);

    // s1.pop();

    // s1.pop();

    // s1.pop();

    // s1.pop();

    // s1.display();

    do

    {

        cout<<"\n0.exit\n1.Push\n2.Pop\n3.Display\n";

        cout<<"Enter choice: ";

        cin>>ch;

        switch (ch)

        {

        case 1:

            if (s1.isFull())

            {

                cout<<"Stack overflow!";

            }else{

                cout<<"Enter element: ";

                cin>>element;

                s1.push(element);

            }

            break;

        case 2:

            s1.pop();

            break;

        case 3:

            s1.display();

            break;

        default:

            cout<<"Invalid";

            break;

        }

    } while (ch!=0);

    return 0;

}

