**Day 1\_\_Lab 1**.

1:write program to test Hello World.

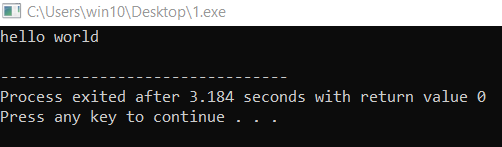
#include<iostream>

using namespace std;

int main(){

cout<<"hello world"<<endl;

}



2:Write a program to adddition of two numbers

#include<iostream>

using namespace std;

int main(){

int a,b,sum;

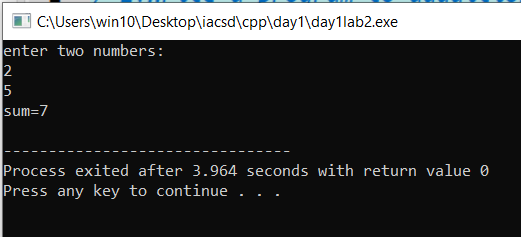
cout<<"enter two numbers:"<<endl;

cin>>a>>b;

sum=a+b;

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

}



3:Write a program to swap two numbers

#include<iostream>

using namespace std;

int main(){

int a,b;

cout<<"enter the number A and B :"<<endl;

cin>>a>>b;

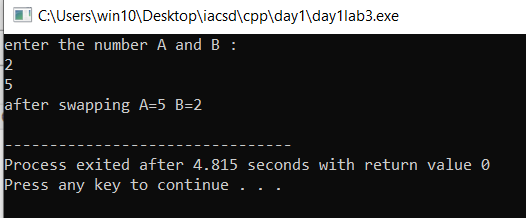
a=a+b;

b=a-b;

a=a-b;

cout<<"after swapping A="<<a<<" B="<<b<<endl;

}



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 the number"<<endl;

cin>>num;

if(num%2==0){

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

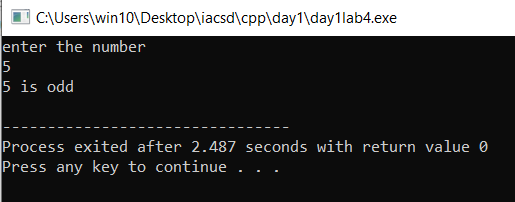
}

else{

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

}

}



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

#include<iostream>

using namespace std;

int main(){

int num;

cout<<"enter the number"<<endl;

cin>>num;

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

cout<<"congrats"<<endl;

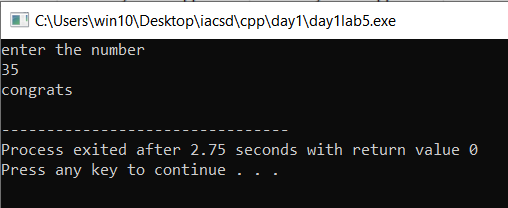
}

else{

cout<<"oopss.."<<endl;

}

}



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;

int main(){

float bs,tax;

cout<<"enter the anual basic salary"<<endl;

cin>>bs;

if(bs<150000){

tax=0;

}

else if(bs>150000 && bs<300000){

tax=.20\*bs;

}

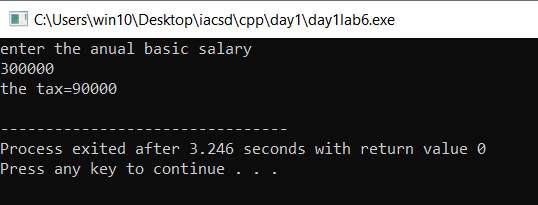
else{

tax=.30\*bs;

}

cout<<"the tax="<<tax<<endl;

}



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 ltr;

cout<<"enter the leter"<<endl;

cin>>ltr;

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

cout<<"vowels"<<endl;

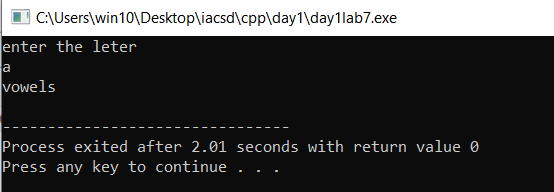
}

else{

cout<<"consonants"<<endl;

}

}



8. 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 a1,a2,a3;

cout<<"enter the angle:"<<endl;

cin>>a1>>a2>>a3;

if(a1+a2+a3==180){

cout<<"triangle is valid"<<endl;

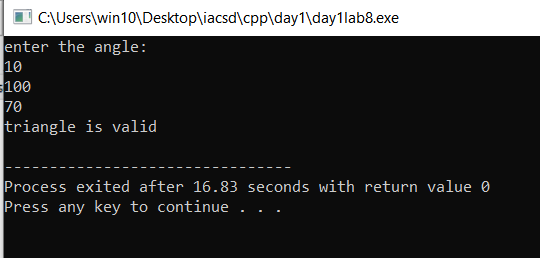
}

else{

cout<<"not valid"<<endl;

}

}



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 num,i;

int fact=1;

cout<<"enter a number"<<endl;

cin>>num;

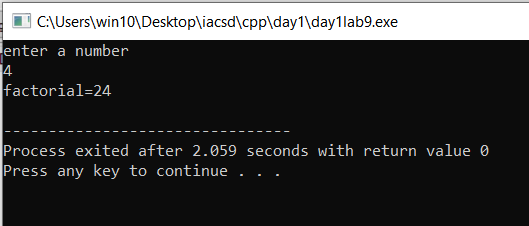
for(i=1;i<=num;i++){

fact=i\*fact;

}

cout<<"factorial="<<fact<<endl;

}



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,i,pwr=1;

cout<<"enter M and N"<<endl;

cin>>m>>n;

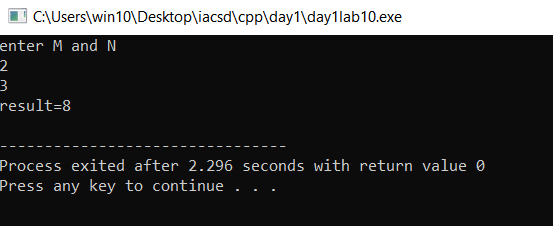
for(i=0;i<n;i++){

pwr=pwr\*m;

}

cout<<"result="<<pwr<<endl;

}



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

#include<iostream>

using namespace std;

int main(){

int num,i;

bool flag;

cout<<"enter number"<<endl;

cin>>num;

for(i=2;i<num;i++){

if(num%i==0){

flag=false;

break;

}

else{

flag=true;

}

}

if(flag){

cout<<"prime"<<endl;

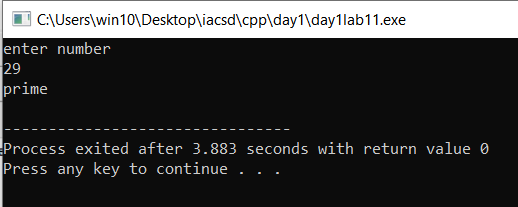
}

else{

cout<<"not prime"<<endl;

}

}



12:Sum of series :1+2+3+….+n

#include<iostream>

using namespace std;

int main()

{

int num;

cout<<"enter number"<<endl;

cin>>num;

int i,sum=0;

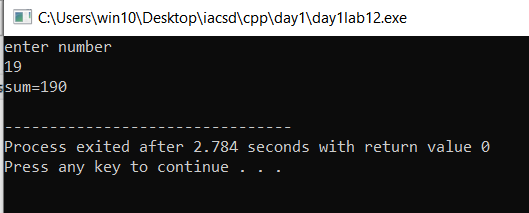
for(i=1;i<=num;i++){

sum=sum+i;

}

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

}



13:Check whether the number is palindrome or not?

#include<iostream>

using namespace std;

int main(){

int n,num,dig,rev;

cout<<"enter the numbr:"<<endl;

cin>>num;

n=num;

while(num!=0){

dig=num%10;

rev=(rev\*10)+dig;

num=num/10;

}

if(n==rev){

cout<<"palindrome"<<endl;

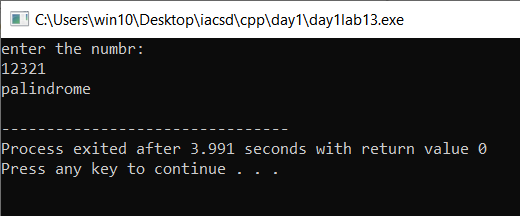
}

else{

cout<<"not palindrome"<<endl;

}

}



14.sum of even and odd

#include<iostream>

using namespace std;

int main(){

int n,i;

cout<<"enter n"<<endl;

cin>>n;

int evenSum=0,oddSum=0;

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

if(i%2==0){

evenSum+=i;

}

else{

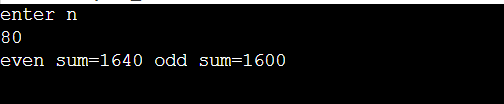
oddSum+=i;

}

}

cout<<"even sum="<<evenSum<<" odd sum="<<oddSum<<endl;

}



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

#include<iostream>

using namespace std;

int main(){

int n,dig,rev=0;

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

cin>>n;

while(n!=0){

dig=n%10;

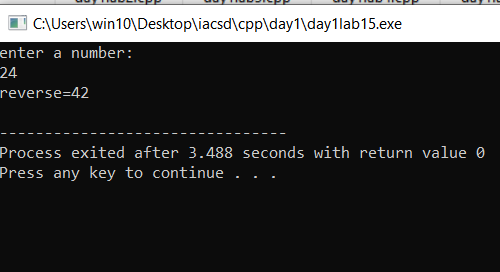
rev=(rev\*10)+dig;

n=n/10;

}

cout<<"reverse="<<rev<<endl;

}



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

#include<iostream>

using namespace std;

int main(){

int i,j,n;

bool flag=false;

cout<<"Enter the number:";

cin>>n;

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

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

if(i%j==0){

flag=false;

break;

}

else{

flag=true;

}

}

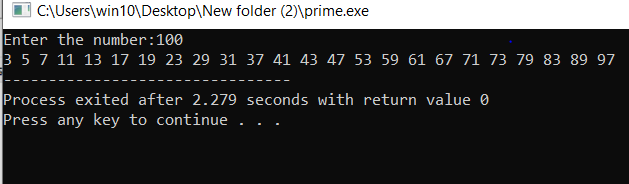
if(flag==true){

cout<<i<<" ";

}

}

}



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

#include<iostream>

#include<cmath>

using namespace std;

int main(){

int num,dig,n1,n2,sum=0;

cout<<"enter the number"<<endl;

cin>>num;

n1=num;

n2=num;

int count=0;

while(n1!=0){

n1=n1/10;

count++;

}

while(n2!=0){

dig=n2%10;

sum=sum+pow(dig,count);

n2=n2/10;

}

if(sum==num){

cout<<"armstrong"<<endl;

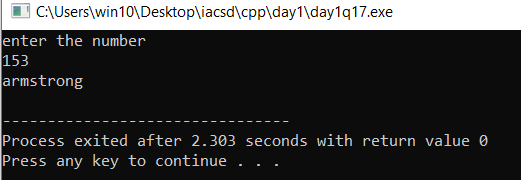
}

else{

cout<<"not armstrong"<<endl;

}

}



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 3 numbers"<<endl;

cin>>a>>b>>c;

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

cout<<"a greater"<<endl;

}

else if(b>a&&b>c){

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

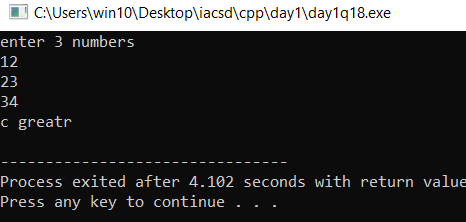
}

else{

cout<<"c greatr"<<endl;

}

}



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

#include<iostream>

using namespace std;

int main(){

int amt,n,i,total=0;

cout<<"the items available: 1 , 2, 3, 4"<<endl;

cout<<"for entering into the shop enter 1"<<endl;

cin>>n;

cout<<"choose your items"<<endl;

while(n!=0){

cin>>n;

switch(n){

case 1:

amt=100;

break;

case 2:

amt=200;

break;

case 3:

amt=300;

break;

case 4:

amt=400;

break;

default:

cout<<"thank you"<<endl;

}

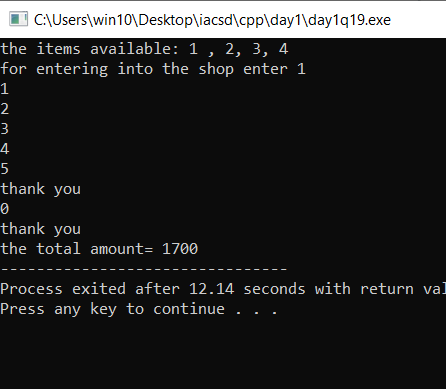
total=total+amt;

}

cout<<"the total amount= "<<total;

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 n;

cout<<"enter the number"<<endl;

cin>>n;

switch(n){

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;

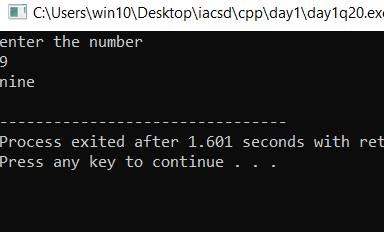
break;

default:

cout<<"oops"<<endl;

}

}



21. 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 a,b,res;

char opt;

cout<<"enter the numbers"<<endl;

cin>>a>>b;

cout<<"enter the operator"<<endl;

cin>>opt;

switch(opt){

case '+':

res=a+b;

break;

case '-':

res=a-b;

break;

case '\*':

res=a\*b;

break;

case '/':

res=a/b;

break;

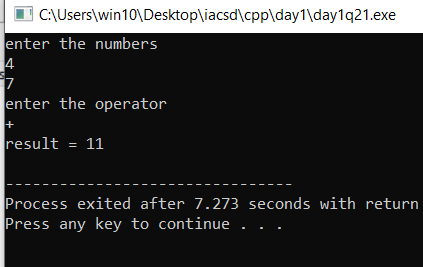
default:

cout<<"wrong operation"<<endl;

}

cout<<"result = "<<res<<endl;

}



**Day 2\_\_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 n,sum=0;

do{

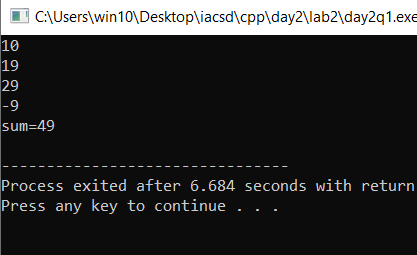
cin>>n;

sum=sum+n;

}while(n>0);

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

}



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,i,res=1;

cout<<"enter x and n"<<endl;

cin>>x>>n;

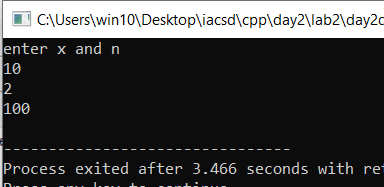
for(i=0;i<n;i++){

res=res\*x;

}

cout<<res<<endl;

}



//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 c;

int n,i;

cout<<"enter char and number"<<endl;

cin>>c>>n;

int N=(int)c;

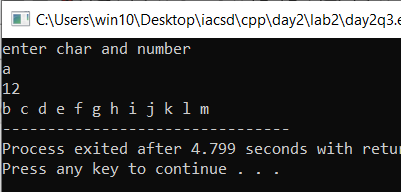
int newN=n+N;

for(i=N+1;i<=newN;i++){

cout<<(char)i<<" ";

}

}



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

#include<iostream>

using namespace std;

int main(){

int n,i,fact=1;

cout<<"enter the number"<<endl;

cin>>n;

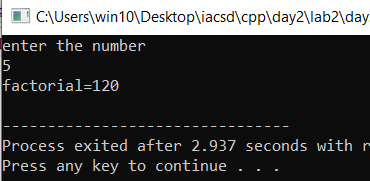
for(i=1;i<=n;i++){

fact=fact\*i;

}

cout<<"factorial="<<fact<<endl;

}



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

#include<iostream>

using namespace std;

int main(){

int num,i;

cout<<"enter the number"<<endl;

cin>>num;

cout<<"factors"<<endl;

for(i=1;i<=num;i++){

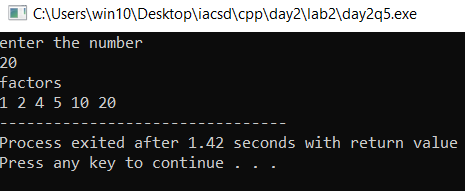
if(num%i==0){

cout<<i<<" ";

}

}

}



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

#include<iostream>

using namespace std;

int main(){

int num1,num2,i,gcd;

cout<<"enter the numbers"<<endl;

cin>>num1>>num2;

for(i=1;i<=num1&&i<=num2;i++){

if(num1%i==0&&num2%i==0){

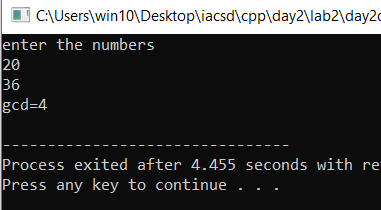
gcd=i;

}

}

cout<<"gcd="<<gcd<<endl;

}



/\*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;

int main(){

int task,area;

cout<<"enter the task"<<endl;

cin>>task;

switch(task){

case 1:

int r;

cout<<"enter the radius"<<endl;

cin>>r;

area=3.14\*r\*r;

break;

case 2:

int l,b;

cout<<"enter length and breadth"<<endl;

cin>>l>>b;

area=l\*b;

break;

case 3:

int h;

cout<<"enter base and height"<<endl;

cin>>b>>h;

area=1/2 \*h\*b;

break;

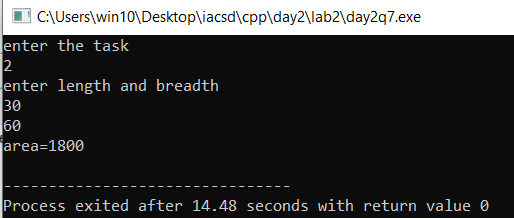
default:

cout<<"exited"<<endl;

}

cout<<"area="<<area<<endl;

}



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

#include<iostream>

using namespace std;

bool isPrime(int n)

{

if (n == 1 || n == 0)

return false;

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

if (n % i == 0)

return false;

}

return true;

}

int main()

{

int N;

cout<<"enter the number:";

cin>>N;

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

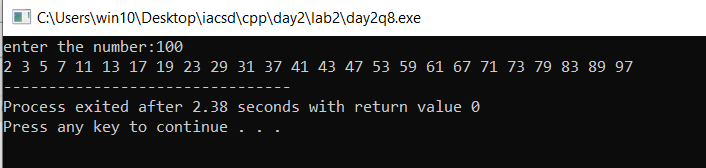
if (isPrime(i))

cout << i << " ";

}

return 0;

}



**Day 2\_\_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 i,size;

cout<<"enter the size"<<endl;

cin>>size;

int arr[size];

cout<<"enter the elemebnts"<<endl;

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

cin>>arr[i];

}

cout<<"sum=";

int sum=0;

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

sum=sum+arr[i];

}

cout<<sum<<endl;

cout<<"average="<<sum/size<<endl;

int temp=arr[0];

for(i=1;i<size;i++){

if(arr[i]>temp){

temp=arr[i];

}

}

cout<<"max="<<temp<<endl;

temp=arr[0];

for(i=1;i<size;i++){

if(arr[i]<temp){

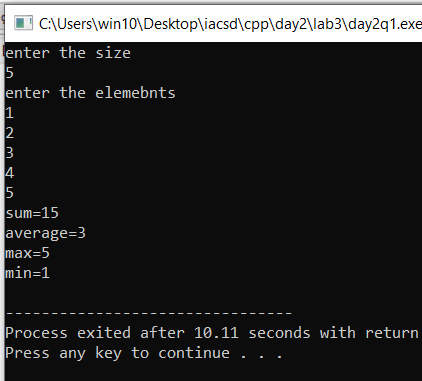
temp=arr[i];

}

}

cout<<"min="<<temp<<endl;

}



//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 n,dig,sum=0;

cout<<"enter number"<<endl;

cin>>n;

while(n!=0){

dig=n%10;

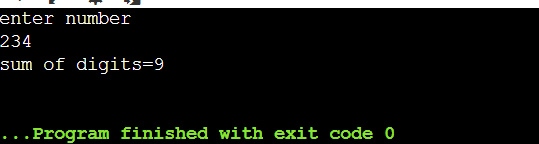
sum=sum+dig;

n=n/10;

}

cout<<"sum of digits="<<sum<<endl;

}



//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,i;

int sumE=0,sumO=0;

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

cin>>n;

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

if(i%2==0){

sumE=sumE+i;

}

else{

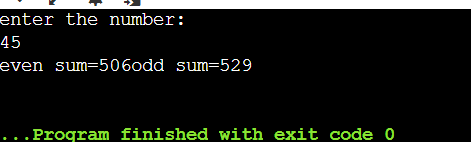
sumO+=i;

}

}

cout<<"even sum="<<sumE<<"odd sum="<<sumO<<endl;

}



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

#include<iostream>

using namespace std;

bool isPrime(int n)

{

if (n == 1 || n == 0)

return false;

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

if (n % i == 0)

return false;

}

return true;

}

int main()

{

int N;

cout<<"enter the number:";

cin>>N;

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

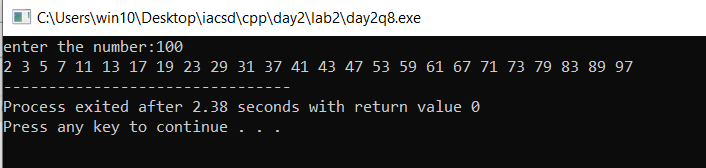
if (isPrime(i))

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 i,num,n;

cout<<"enter size"<<endl;

cin>>n;

int arr[n];

cout<<"enter elements"<<endl;

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

cin>>arr[i];

}

cout<<"enter the number to be searched"<<endl;

cin>>num;

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

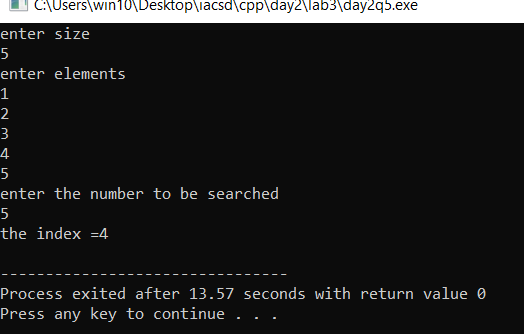
if(num==arr[i]){

cout<<"the index ="<<i<<endl;

}

}

}



/\* 6:Write a program to print following pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\*/

#include<iostream>

using namespace std;

int main(){

int i,j,n;

cout<<"enter num="<<endl;

cin>>n;

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

for(j=0;j<=i;j++){

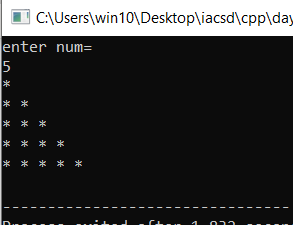
cout<<"\* ";

}

cout<<endl;

}

}



/\*7: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>

using namespace std;

class student{

private: int rollno;

int m1,m2,m3;

float total,perc;

char grade;

public: void acceptinfo(){

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

cin>>rollno;

cout<<"enter the marks:"<<endl;

cin>>m1>>m2>>m3;

total=(float)(m1+m2+m3);

perc=total/100 \*100;

if(perc>80){

grade='A';

}

else if(perc>60 && perc<80){

grade='B';

}

else{

grade='C';

}

}

void display(){

cout<<"student details"<<endl;

cout<<"roll number="<<rollno<<endl;

cout<<"total marks="<<total<<endl;

cout<<"percentage="<<perc<<endl;

cout<<"grade="<<grade<<endl;

}

};

int main(){

student obj;

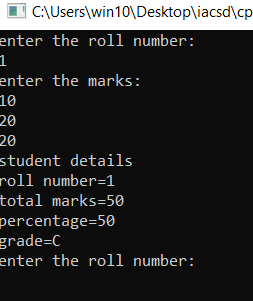
obj.acceptinfo();

obj.display();

student obj2;

obj2.acceptinfo();

}



**Day3\_lab4**.

/\*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>

using namespace std;

class student{

private: int rollno;

int m1,m2,m3;

double total;

float perc;

char grade;

public: void acceptinfo(){

cout<<"enter the student details"<<endl;

cout<<"enter the roll number"<<endl;

cin>>rollno;

cout<<"enter the marks:"<<endl;

cin>>m1>>m2>>m3;

}

void display(){

cout<<"student details"<<endl;

cout<<"roll number="<<rollno<<endl;

cout<<"total marks="<<total<<endl;

cout<<"percentage="<<perc<<endl;

cout<<"grade="<<grade<<endl;

}

void Total(){

total=m1+m2+m3;

}

void percentage(){

perc=(total/300)\*100;

}

void Grade(){

if(perc>=80){

grade='A';

}

else if(perc<80 && perc>=60){

grade='B';

}

else{

grade='C';

}

}

};

int main(){

student obj1;

obj1.acceptinfo();

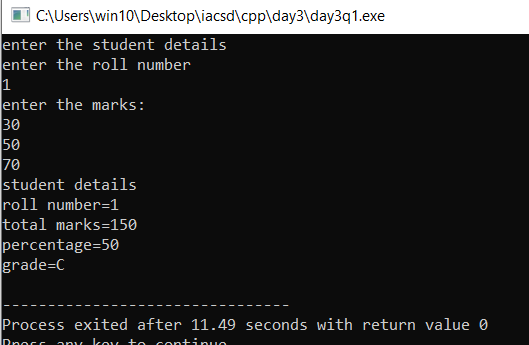
obj1.Total();

obj1.percentage();

obj1.Grade();

obj1.display();

}



/\*2. 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{

private :char \* name;

int age;

char \* city;

public:Person(){ //default constructor

this->name=new char;

\*name='\0';

this->age=18;

this->city=new char;

\*city='\0';

cout<<"person get created(default constr)"<<endl;

}

Person(char \* name,int age,char \* city){ //parameterised constructor

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

this->age=age;

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

cout<<"Person get created(parameterised constr)"<<endl;

}

void assigndetails(char \* name,int age,char \* city){

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

strcpy(this->name,name);

this->age=age;

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

strcpy(this->city,city);

}

void display(){

cout<<" name=";

for(int i=0;i<strlen(this->name);i++){

cout<<\*(name+i);

}

cout<<" city=";

for(int j=0;j<strlen(this->city);j++){

cout<<\*(city+j);

}

cout<<" age="<<age<<endl;;

}

void setName(char\* name){

strcpy(this->name,name);

}

void setAge(int age){

this->age=age;

}

void setCity(char\* city){

strcpy(this->city,city);

}

char\* getName(){

return this->name;

}

int getAge(){

return this->age;

}

char\* getCity(){

return this->city;

}

};

int main(){

Person obj1;

Person obj2("adheesh",24,"pune");

obj2.assigndetails("adheesh",24,"pune");

obj2.display();

obj1.assigndetails("anu",23,"chennai");

obj1.display();

obj1.setName("anagha");

obj1.setAge(24);

obj1.setCity("kasargod");

obj1.display();

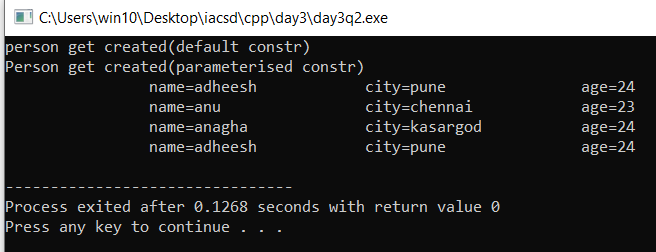
obj2.getName();

obj2.getAge();

obj2.getCity();

obj2.display();

}



/\*3. 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 day,month,year;

public :Date(){

day=01;

month=01;

year=2024;

}

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

this->day=day;

this->month=month;

this->year=year;

}

void acceptdate(int day,int month,int year){

this->day=day;

this->month=month;

this->year=year;

}

void display(){

cout<<"Date : "<<day<<"/"<<month<<"/"<<year<<endl;

}

void setDay(int day){

this->day=day;

}

void setMonth(int month){

this->month=month;

}

void setYear(int year){

this->year=year;

}

int getDay(){

return this->day;

}

int getMonth(){

return this->month;

}

int getYear(){

return this->year;

}

};

int main(){

Date d1;

Date d2(03,02,2000);

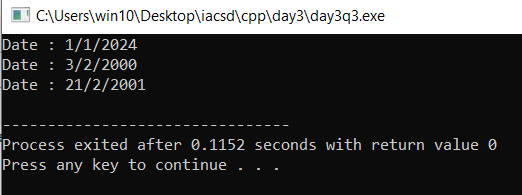
d1.display();

d2.display();

d1.acceptdate(21,02,2001);

d1.display();

}



/\*4.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{

private :string bname;

int id;

string author;

int price;

public :Book(){

this->bname="N/A";

this->id=0;

this->author="N/A";

this->price=0;

cout<<"default book details"<<endl;

}

Book(string bname,int id,string author,int price){

this->bname=bname;

this->id=id;

this->author=author;

this->price=price;

cout<<"parameterised details of book"<<endl;

}

void acceptdetails(string bname,int id,string author,int price){

this->bname=bname;

//strcpy(this->bname,bname);

this->id=id;

this->author=author;

// strcpy(this->author,author);

this->price=price;

}

void display(){

cout<<"the entered details of book:"<<endl;

cout<<"book name :"<<bname<<endl;

// for(int i=0;i<strlen(this->bname);i++){

// cout<<\*(bname+i);

// }

cout<<""<<endl;

cout<<"book id :"<<id<<endl;

cout<<"book author :"<<author<<endl;

// for(int j=0;j<strlen(this->author);j++){

// cout<<\*(author+j);

// }

cout<<""<<endl;

cout<<"price :"<<price;

}

};

int main(){

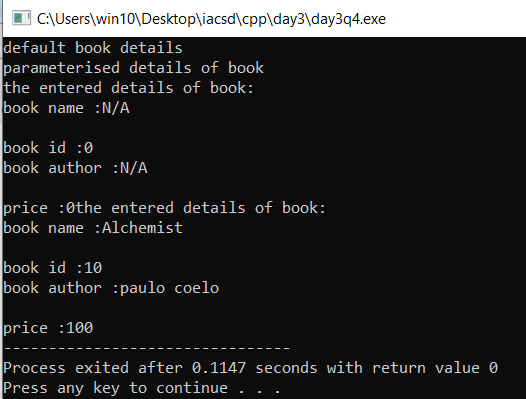
Book b1;

Book b2("Alchemist",10,"paulo coelo",100);

b1.display();

b2.display();

}



/\*5. 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;

class Point{

private: int x,y;

public: Point(){

x=1;

y=1;

cout<<"default point"<<endl;

}

Point(int x,int y){

this->x=x;

this->y=y;

cout<<"parameter point"<<endl;

}

void acceptPoint(int x,int y){

this->x=x;

this->y=y;

}

void display(){

cout<<"the points are : ("<<this->x<<" , "<<this->y<<")"<<endl;

}

void setX(int x){

this->x=x;

}

void setY(int y){

this->y=y;

}

int getX(){

return this->x;

}

int getY(){

return this->y;

}

};

int main(){

Point p1;

Point p2(10,10);

p1.display();

p2.display();

p1.acceptPoint(20,20);

p2.acceptPoint(30,30);

int a=p1.getX();

int b=p2.getY();

cout<<a<<" "<<b;

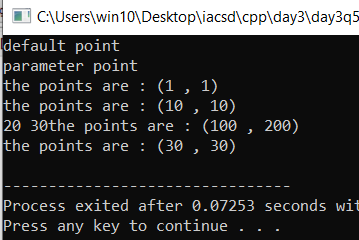
p1.setX(100);

p1.setY(200);

p1.display();

p2.display();

}



/\*6. 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{

private: int i,r;

public: complexNumber(){

i=1;

r=1;

}

complexNumber(int i,int r){

this->i=i;

this->r=r;

cout<<"default"<<endl;

}

void setI(int i){

this->i=i;

}

void setR(int r){

this->r=r;

}

int getI(){

return this->i;

}

int getR(){

return this->r;

}

void acceptNumber(int i,int r){

this->i=i;

this->r=r;

}

void display(){

cout<<"the complex number="<<i<<"!"<<"+"<<r<<endl;

}

};

int main(){

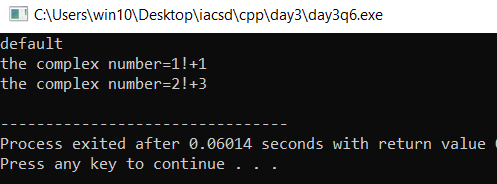
complexNumber obj1;

complexNumber obj2(2,3);

obj1.display();

obj2.display();

}



**Day4\_inheritance lab**

1. 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;

#ifndef ANIMAL\_H

#define ANIMAL\_H

class Animal{

private:

string name;

int lifespan;

string region;

public:

void acceptDetails(){

cout<<"Enter the animal details"<<endl;

cout<<"Name:";

cin>>name;

cout<<"\nLifespan:";

cin>>lifespan;

cout<<"\nRegion:";

cin>>region;

cout<<name<<" Details are added:"<<endl;

}

void display(){

cout<<name<<" details-----"<<endl;

cout<<"NAME of animal:"<<name<<endl;

cout<<"LIFESPAN of animal:"<<lifespan<<endl;

cout<<"REGION of animal:"<<region<<endl;

}

};

#endif

#include"Animal.h"

using namespace std;

class Mamal:public Animal{

private:

int NoLegs;

int NoBreast;

public:

void PhysicalApperence(){

cout<<"Enter No of Legs:";

cin>>NoLegs;

cout<<"\nEnter No of Breast:";

cin>>NoBreast;

}

void display(){

Animal::display();

cout<<"Number of legs:"<<NoLegs<<endl;

cout<<"Number of breast:"<<NoBreast<<endl;

}

};

#include"Animal.h"

using namespace std;

class Fish:public Animal{

private:

int length;

int weight;

string habitate;

public:

void size(){

cout<<"Enter the lenghth of fish:";

cin>>length;

cout<<"\nEnter the weight of fish:";

cin>>weight;

}

void LivingHabitate(){

cout<<"Enter the habitate:";

cin>>habitate;

}

void display(){

Animal::display();

cout<<"Length of fish:"<<length<<endl;

cout<<"Weight of fish:"<<weight<<endl;

cout<<"Habitate:"<<habitate<<endl;

}

};

#include"Animal.h"

using namespace std;

class Bird:public Animal{

private:

string color;

public:

void Charecterstics(){

cout<<"Enter the color:";

cin>>color;

cout<<endl;

}

void display(){

Animal::display();

cout<<"Color:"<<color<<endl;

}

};

#include<iostream>

#include"Animal.h"

#include"Mamal.h"

#include"Fish.h"

#include"Bird.h"

using namespace std;

int main(){

Animal a1;

Fish f1;

Mamal m1;

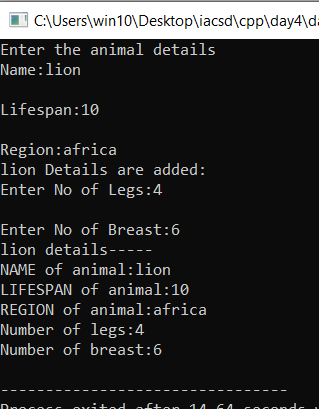
Bird b1;

m1.acceptDetails();

m1.PhysicalApperence();

m1.display();

}



2.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.

\*/

#include<iostream>

using namespace std;

#ifndef SHAPE\_H

#define SHAPE\_H

class Shape{

public:

virtual void area()=0;

virtual void perimeter()=0;

};

#endif

#include<iostream>

#include"Shape.h"

using namespace std;

class Circle:public Shape{

private:

int r,A,peri;

public:

void area(){

cout<<"enter the radius:";

cin>>r;

A=3.14\*r\*r;

cout<<"Area of circle="<<A<<endl;

}

void perimeter(){

peri=2\*3.14\*r;

cout<<"Perimeter of circle="<<peri<<endl;

}

};

#include<iostream>

#include"Shape.h"

using namespace std;

class Rectangle:public Shape{

private:

int l,b,A,peri;

public:

void area(){

cout<<"enter the length and breadth:";

cin>>l>>b;

A=l\*b;

cout<<"Area of rectangle="<<A<<endl;

}

void perimeter(){

cout<<"enter the length and breadth:";

cin>>l>>b;

peri=2\*(l+b);

cout<<"Perimeter of rectangle="<<peri<<endl;

}

};

#include<iostream>

#include<math.h>

#include"Shape.h"

using namespace std;

class Triangle:public Shape{

private:

float b,h,x,A,peri;

public:

void area(){

cout<<"enter the base and height:";

cin>>b>>h;

A=0.5\*(b\*h);

cout<<"Area of triangle="<<A<<endl;

}

void perimeter(){

cout<<"enter the base , height and hypotneous:";

cin>>b>>h>>x;

peri=x+b+h;

cout<<"Perimeter of triangle="<<peri<<endl;

}

};

#include<iostream>

#include"Shape.h"

#include"Circle.h"

#include"Rectangle.h"

#include"Triangle.h"

using namespace std;

int main(){

Circle c;

c.area();

c.perimeter();

Rectangle R;

R.area();

R.perimeter();

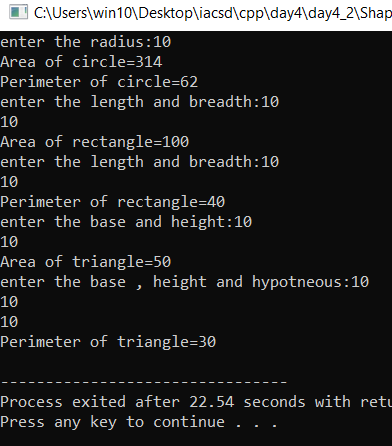
//

Triangle t;

t.area();

t.perimeter();

}



3.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;

#ifndef EMPLOYEE\_H

#define EMPLOYEE\_H

class Employee{

private:

int empid;

double salary;

public:

void acceptDetails(){

cout<<"enter the employee details---"<<endl;

cout<<"enter employee id:";

cin>>empid;

cout<<"enter salary:";

cin>>salary;

cout<<"employee details are added!!!"<<endl;

}

void calcSalary(){

cout<<"salary="<<salary<<endl;

}

double getSalary(){

return salary;

}

};

#endif

#include<iostream>

#include"Employee.h"

using namespace std;

class Manager:public Employee{

private:

int deptid;

double incentive;

public:

void ManagerTask(){

cout<<"manager task done"<<endl;

}

};

#include<iostream>

#include"Employee.h"

using namespace std;

class Developer:public Employee{

private:

int projId;

double salary,bonus;

public:

void project(){

cout<<"enter the project id:";

cin>>projId;

if(projId!=0){

cout<<"enter the bonus for developer:"<<endl;

cin>>this->bonus;

cout<<"developer is workin in project with id:"<<projId<<endl;

}

else{

this->bonus=0;

cout<<"developer has no project now ,So no bonus applied"<<endl;

}

}

};

#include<iostream>

#include"Employee.h"

#include"Manager.h"

#include"Developer.h"

using namespace std;

int main(){

Manager m;

m.acceptDetails();

m.calcSalary();

m.ManagerTask();

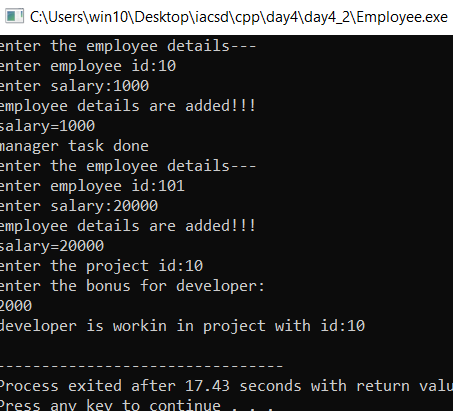
Developer d;

d.acceptDetails();

d.calcSalary();

d.project();

}



4. /\*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;

#ifndef VEHICLE\_H

#define VEHICLE\_H

class Vehicle{

private:

string category;

public:

Vehicle(){

category="None";

cout<<"default vehicle"<<endl;

}

Vehicle(string category){

this->category=category;

//cout<<"parameterised vehicle"<<endl;

}

void acceptDetails(){

cout<<"Enter the category:";

cin>>this->category;

cout<<"details updated"<<endl;

}

// virtual void startEngine()=0;

// virtual void Wheel()=0;

//

virtual void display(){

cout<<"Category: "<<this->category<<endl;

}

};

#endif

#include<iostream>

#include"Vehicle.h"

using namespace std;

class Car:public Vehicle{

private:

int wheelno;

// string engine;

public:

Car(){

wheelno=0;

//engine="None";

cout<<"default car"<<endl;

}

Car(string category,int wheelno):Vehicle(category){

this->wheelno=wheelno;

}

void startEngine(string engine){

cout<<"Engine type: "<<engine<<endl;

}

void display(){

Vehicle::display();

cout<<"Number of wheels:"<<this->wheelno<<endl;

}

};

#include<iostream>

#include"Vehicle.h"

using namespace std;

class Bike:public Vehicle{

private:

// int wheelno;

string engine;

public:

Bike(){

//wheelno=0;

engine="None";

cout<<"default bike"<<endl;

}

Bike(string category,string engine):Vehicle(category){

this->engine=engine;

}

void Wheel(int wheelno){

cout<<"Number of wheels: "<<wheelno<<endl;

}

void display(){

Vehicle::display();

cout<<"Engine type:"<<this->engine<<endl;

}

};

#include<iostream>

#include"Vehicle.h"

using namespace std;

class Truck:public Vehicle{

private:

int wheelno;

string engine;

public:

Truck(){

wheelno=0;

engine="None";

cout<<"default truck"<<endl;

}

Truck(string category,int wheelno):Vehicle(category){

this->wheelno=wheelno;

//cout<<"parameterised truck"<<endl;

}

void startEngine(string engine){

cout<<"Engine type: "<<engine<<endl;

}

void display(){

Vehicle::display();

cout<<"Number of wheels: "<<this->wheelno<<endl;

}

//

};

#include<iostream>

#include"Vehicle.h"

#include"Car.h"

#include"Bike.h"

#include"Truck.h"

#include<typeinfo>

using namespace std;

int main(){

Vehicle \*fun[4];

Car c1("Car",4);

Bike b1("Bike","petrol");

Truck t1("Truck",8);

fun[0]=&c1;

fun[1]=&b1;

fun[2]=&t1;

for(int i=0;i<3;i++){

fun[i]->display();

if(typeid(\*fun[i])==typeid(Car)){

Car\*c=dynamic\_cast<Car\*>(fun[i]);

c->startEngine("petrol");

}

if(typeid(\*fun[i])==typeid(Bike)){

Bike\*b=dynamic\_cast<Bike\*>(fun[i]);

b->Wheel(2);

}

// if(typeid(\*fun[i])==typeid(Truck)){

// Truck\*t=dynamic\_cast<Truck\*>(fun[i]);

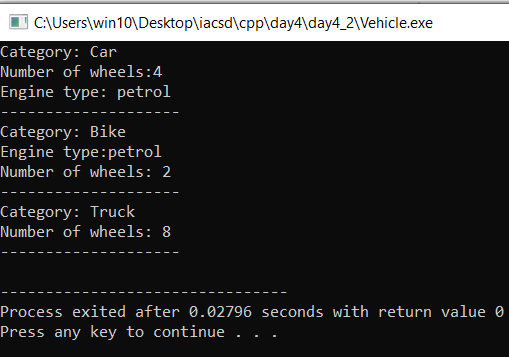
// t->startEngine("Diesel");

// }

cout<<"--------------------"<<endl;

}

}



5. /\*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;

#ifndef BANK\_H

#define BANK\_H

class Bankaccount{

private:

int accNO;

double balance;

public:

Bankaccount(){

accNO=0;

balance=0;

}

Bankaccount(int accNo,double balance){

this->accNO=accNO;

if(balance>=5000){

this->balance=balance;

}

else{

cout<<"Minimum balance is 5000!!!"<<endl;

}

}

void acceptDetails(){

cout<<"Enter the account number:";

cin>>accNO;

}

virtual void Withdraw(){

cout<<"withhhhh"<<endl;

}

virtual void calcInterest(){

cout<<"intrrrss"<<endl;

}

};

#endif

#include<iostream>

#include"Bankaccount.h"

using namespace std;

class Savingsaccount:public Bankaccount{

private:

double savings,amt,bal;

float interest;

public:

Savingsaccount(){

savings=0;

interest=0;

}

Savingsaccount(int accNO,double balance,double savings,float interest):Bankaccount(accNO,balance){

this->savings=savings;

this->interest=interest;

}

void Withdraw(){

cout<<"savings withdraw"<<endl;

}

void calcInterest(){

cout<<"savings interest"<<endl;

}

void display(){

cout<<"savings display"<<endl;

}

};

#include<iostream>

#include"Bankaccount.h"

using namespace std;

class Checkingaccount:public Bankaccount{

int accno;

public:

Checkingaccount(int accno){

this->accno=accno;

cout<<"\n\n\nHello user"<<endl;

}

void Withdraw(){

cout<<"checking withdraw"<<endl;

}

void calcInterst(){

cout<<"checking interest"<<endl;

}

void display(){

cout<<"checking display"<<endl;

}

};

#include<iostream>

#include"Bankaccount.h"

#include"Savingsaccount.h"

#include"Checkingaccount.h"

using namespace std;

int main(){

Savingsaccount s(101,1000,20000,3.5);

s.Withdraw();

s.calcInterest();

s.display();

// Checkingaccount (500);

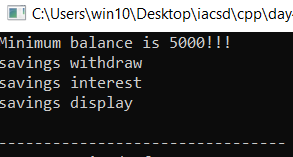
//

// c.Withdraw();

//

// c.display();

}



6. /\*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;

#ifndef SHAPE\_H

#define SHAPE\_H

class GeometricShape{

private:

int area,perimeter;

string shapename;

public:

Shape(){

area=0;

perimeter=0;

shapename="Shapeless";

}

void Area(){

cout<<"Area of shape"<<endl;

}

void Perimeter(){

cout<<"perimeter of shape"<<endl;

}

};

#endif

#include<iostream>

#include"GeometricShape.h"

using namespace std;

class GeometricCircle:public GeometricShape{

int r,peri,area;

public:

void Area(){

cout<<"enter the radius:";

cin>>r;

area=3.14\*r\*r;

cout<<"Area="<<area<<endl;

}

void Perimeter(){

cout<<"enter the radius:";

cin>>r;

peri=2\*3.14\*r;

cout<<"Perimeter="<<peri<<endl;

}

};

#include<iostream>

#include"GeometricShape.h"

using namespace std;

class GeometricRectangle:public GeometricShape{

int l,b,area,peri;

public:

void Area(){

cout<<"Enter the length and breadth:";

cin>>l>>b;

area=l\*b;

cout<<"Area="<<area<<endl;

}

void Perimeter(){

cout<<"Enter the length and breadth:";

cin>>l>>b;

peri=2\*(l+b);

cout<<"Perimeter="<<peri<<endl;

}

};

#include<iostream>

#include"GeometricShape.h"

using namespace std;

class GeometricTriangle:public GeometricShape{

int b,h,x,area,peri;

public:

void Area(){

cout<<"Enter the height , base and hypotneous:";

cin>>h>>b>>x;

area=0.5\*b\*h;

cout<<"Area="<<area<<endl;

}

void Perimeter(){

cout<<"Enter the height , base and hypotneous:";

cin>>h>>b>>x;

peri=b+h+x;

cout<<"Perimeter="<<peri<<endl;

}

};

#include<iostream>

#include"GeometricShape.h"

#include"GeometricCircle.h"

#include"GeometricRectangle.h"

#include"GeometricTriangle.h"

using namespace std;

int main(){

GeometricCircle c;

GeometricRectangle r;

GeometricTriangle t;

c.Area();

c.Perimeter();

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

r.Area();

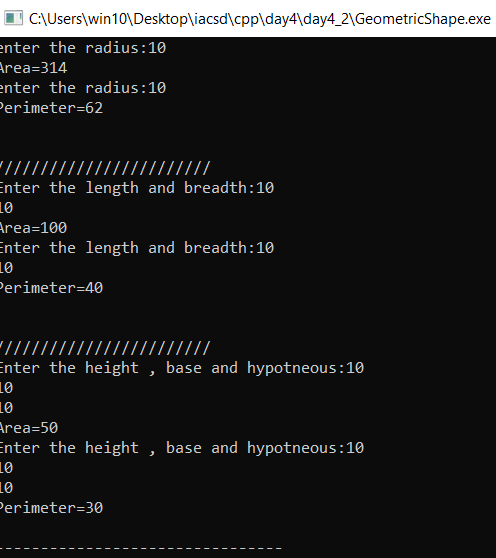
r.Perimeter();

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

t.Area();

t.Perimeter();

}



7. /\*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;

#ifndef PERSON\_H

#define PERSON\_H

class Person{

private:

string name;

int age;

public:

Person(){

name="NONE";

age=0;

}

Person(string name,int age){

this->name=name;

this->age=age;

cout<<"Parameterised Person/////"<<endl;

}

void acceptDetails(){

cout<<"Enter the name:";

cin>>this->name;

cout<<"Enter the age:";

cin>>this->age;

}

void display(){

cout<<"NAME: "<<this->name<<endl;

cout<<"AGE: "<<this->age<<endl;

}

string getName(){

return this->name;

}

int getAge(){

return this->age;

}

};

#endif

#include<iostream>

#include"Person.h"

using namespace std;

class Student:public Person{

private:

int studId;

float gpa;

public:

Student(){

studId=0;

gpa=0.0;

}

Student(string name,int age,int studId,float gpa):Person(name,age){

this->studId=studId;

this->gpa=gpa;

cout<<"Parameterised Student/////"<<endl;

}

void studInput(){

acceptDetails();

cout<<"Enter the student ID:";

cin>>studId;

cout<<"Enter the GPA:";

cin>>gpa;

cout<<"Student details are added/////"<<endl;

}

void display(){

Person::display();

cout<<"STUDENT ID: "<<studId<<endl;

cout<<"GPA: "<<gpa<<endl;

}

};

#include<iostream>

#include"Person.h"

#include"Student.h"

using namespace std;

int main(){

Student s("Adheesh",24,3,6.9);

s.display();

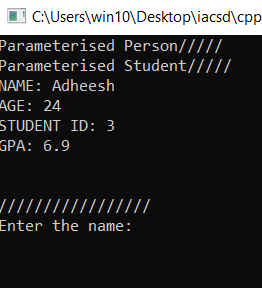
cout<<"\n\n/////////////////"<<endl;

Student s1;

s1.studInput();

s1.display();

}



8. /\*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;

#ifndef LIBRARY\_H

#define LIBRARY\_H

class Library{

private:

int itemId;

string title,author;

string member;

public:

Library(){

itemId=0;

title="None";

author="None";

member="None";

}

Library(int itemId,string title,string author,string member){

this->itemId=itemId;

this->title=title;

this->author=author;

this->member=member;

cout<<"Parametrised Library/////"<<endl;

}

void ItemDetails(){

cout<<"Enter the item ID:";

cin>>itemId;

cout<<"Enter the title:";

cin>>title;

cout<<"Enter the author name:";

cin>>author;

cout<<"Enter the name of person who taking book:";

cin>>member;

}

void display(){

cout<<"Member: "<<member<<endl;

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

cout<<"Item ID: "<<itemId<<endl;

cout<<"Author: "<<author<<endl;

}

void date(int d,int m,int y){

cout<<d<<"/"<<m<<"/"<<y;

}

};

#endif

#include<iostream>

#include"Library.h"

using namespace std;

class Book:public Library{

private:

int dc,mc,yc,dr,mr,yr;

public:

Book(){

dc=0;mc=0;yc=0;

}

Book(int itemId,string title,string author,string member,int dc,int mc,int yc):Library(itemId,title,author,member){

this->dc=dc;

this->mc=mc;

this->yc=yc;

cout<<"Parameterised Book/////"<<endl;

}

void Checkout(){

cout<<"Enter the checkout date:";

cin>>dc>>mc>>yc;

}

void Return(){

cout<<"Enter the return date:";

cin>>dr>>mr>>yr;

}

void display(){

Library::display();

cout<<"Checkout date: ";

Library::date(dc,mc,yc);

cout<<"\n"<<endl;

cout<<"Return date: ";

Library::date(dr,mr,yr);

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

}

};

#include<iostream>

#include"Library.h"

using namespace std;

class Journal:public Library{

private:

int dc,mc,yc,dr,mr,yr;

public:

Journal(){

dc=0;mc=0;yc=0;

}

Journal(int itemId,string title,string author,string member,int dc,int mc,int yc):Library(itemId,title,author,member){

this->dc=dc;

this->mc=mc;

this->yc=yc;

cout<<"Parameterised Journal/////"<<endl;

}

void Checkout(){

cout<<"Enter the checkout date:";

cin>>dc>>mc>>yc;

}

void Return(){

cout<<"Enter the return date:";

cin>>dr>>mr>>yr;

}

void display(){

Library::display();

cout<<"Checkout date: ";

date(dc,mc,yc);

cout<<"Return date: ";

date(dr,mr,yr);

}

};

#include<iostream>

#include"Library.h"

#include"Book.h"

#include"Journal.h"

using namespace std;

int main(){

Book b;

b.ItemDetails();

b.Checkout();

b.Return();

b.display();

Journal j;

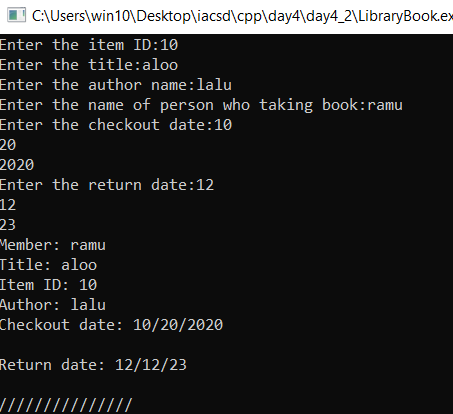
j.ItemDetails();

j.Checkout();

j.Return();

j.display();

}



9. /\*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>

#include <algorithm>

#include <math.h>

using namespace std;

class Shape {

public:

virtual double area() const = 0;

virtual double perimeter() const = 0;

};

class Circle : public Shape {

private:

double radius;

public:

Circle(double radius) : radius(radius) {}

double area() const {

return 3.14159 \* radius \* radius;

}

double perimeter() const override {

return 2 \* 3.14159 \* radius;

}

};

class Rectangle : public Shape {

private:

double length, width;

public:

Rectangle(double length, double width) : length(length), width(width) {}

double area() const override {

return length \* width;

}

double perimeter() const override {

return 2 \* (length + width);

}

};

class Triangle : public Shape {

private:

double side1, side2, side3;

public:

Triangle(double side1, double side2, double side3) : side1(side1), side2(side2), side3(side3) {}

double area() const override {

double s = (side1 + side2 + side3) / 2;

return sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

double perimeter() const override {

return side1 + side2 + side3;

}

};

bool compareShapesByArea(const Shape\* shape1, const Shape\* shape2) {

return shape1->area() < shape2->area();

}

int main() {

Circle circle(5);

Rectangle rectangle(4, 6);

Triangle triangle(3, 4, 5);

Shape\* shapes[] = {&circle, &rectangle, &triangle};

int size = sizeof(shapes) / sizeof(shapes[0]);

sort(shapes, shapes + size, compareShapesByArea);

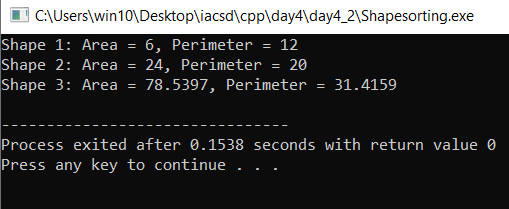
for (int i = 0; i < size; i++) {

cout << "Shape " << i + 1 << ": Area = " << shapes[i]->area() << ", Perimeter = " << shapes[i]->perimeter() << endl;

}

return 0;

}



**Day5\_lab6.**

1. /\*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\*/

Answer:

#include<iostream>

using namespace std;

#ifndef EMP\_H

#define EMP\_H

class EMP{

private:

int id;

string name;

int deptId;

double basicSalary;

public:

EMP(){

id=0;

name="None";

deptId=0;

basicSalary=0;

}

EMP(int id,string name,int deptId,double basicSalary){

this->id=id;

this->name=name;

this->deptId=deptId;

this->basicSalary=basicSalary;

cout<<"parameterised EMP---"<<endl;

}

void acceptDetails(){

cout<<"enter the employee ID:";

cin>>id;

cout<<"enter the name:";

cin>>name;

cout<<"Dept ID:";

cin>>deptId;

cout<<"enter the basic salary:";

cin>>basicSalary;

}

double computeNetSalary(){

cout<<"Net salary of employee="<<basicSalary;

}

double getSalary(){

return this->basicSalary;

}

void display(){

cout<<"EMP ID-> "<<id<<endl;

cout<<"NAME-> "<<name<<endl;

cout<<"DEPT ID-> "<<deptId<<endl;

cout<<"BASIC SALARY-> "<<basicSalary<<endl;

}

};

#endif

#include<iostream>

#include"EMP.h"

using namespace std;

class MGR:public EMP{

private:

double perfBonus;

double basic;

public:

MGR(){

perfBonus=0;

}

MGR(int id,string name,int deptId,double basicSalary,double perfBonus):EMP(id,name,deptId,basicSalary){

this->perfBonus=perfBonus;

cout<<"parameterised MGR---"<<endl;

}

void computeNetSalary(){

cout<<"enter the bonus for manager="<<endl;

cin>>perfBonus;

basic=getSalary()+perfBonus;

cout<<"Net salary of manager="<<basic<<endl;

}

void acceptDetails(){

EMP::acceptDetails();

cout<<"enter the perf Bonus:";

cin>>perfBonus;

}

void display(){

EMP::display();

cout<<"PERF BONUS-> "<<perfBonus<<endl;

cout<<"NET SALARY-> "<<basic<<endl;

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

}

};

#include<iostream>

#include"EMP.h"

using namespace std;

class WRKR:public EMP{

private:

int hoursWorked;

double hourlyRate;

double basic;

public:

WRKR(int id,string name,int deptId,double basicSalary,int hoursWorked,double hourlyRate):EMP(id,name,deptId,basicSalary){

this->hoursWorked=hoursWorked;

this->hourlyRate=hourlyRate;

cout<<"parameterised WRKR---"<<endl;

}

void computeNetSalary(){

basic=getSalary()+(this->hoursWorked\*this->hourlyRate);

cout<<"Net salary of worker="<<basic<<endl;

}

void acceptDetails(){

EMP::acceptDetails();

cout<<"enter the hours worked by the worker="<<endl;

cin>>this->hoursWorked;

cout<<"enter the hourly rate of worker="<<endl;

cin>>this->hourlyRate;

}

double getHourlyRate(){

return (this->hoursWorked\*this->hourlyRate);

}

};

#include<iostream>

#include"EMP.h"

#include"MGR.h"

#include"WRKR.h"

using namespace std;

int main(){

MGR \*mgr[5];

MGR m1,m2,m3,m4,m5;

mgr[0]=&m1;

mgr[1]=&m2;

mgr[2]=&m3;

mgr[3]=&m4;

mgr[4]=&m5;

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

mgr[i]->acceptDetails();

}

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

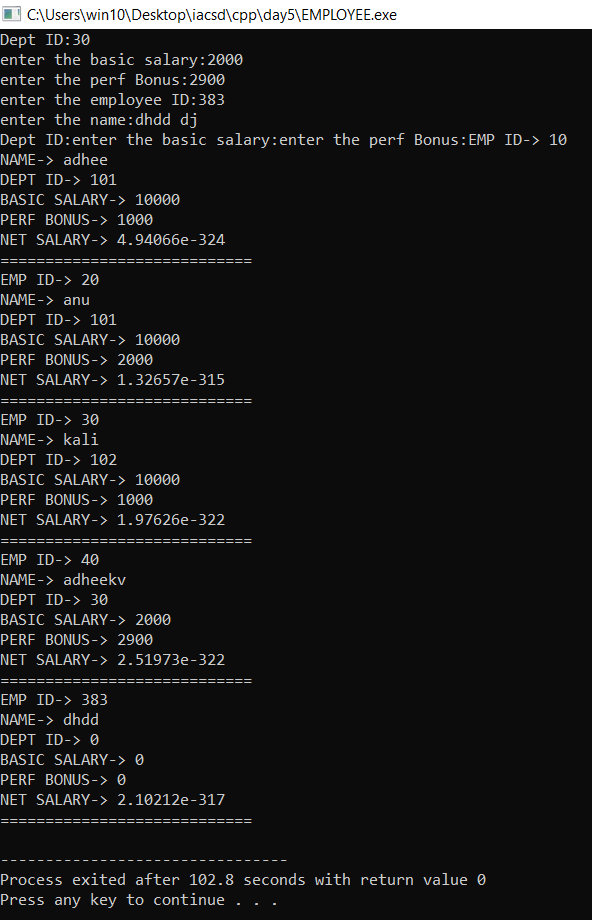
mgr[j]->display();

}

//mgr[0]->display();

return 0;

}



2. /\*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{

private:

int accNo;

string custName;

double balance;

public:

BankAccount(){

accNo=0;

custName="None";

balance=0;

}

BankAccount(int accNo,string custName,double balance){

this->accNo=accNo;

this->custName=custName;

this->balance=balance;

}

void CreateAccount(){

cout<<"Enter the name:";

cin>>custName;

cout<<"Enter account number:";

cin>>accNo;

cout<<"Minimum balance is 5000!"<<endl;

}

void deposite(int amount){

this->balance=this->balance+amount;

}

void withdraw(int amount){

double am=this->balance-amount;

if(am>=5000){

this->balance=this->balance-amount;

}

}

void display(){

cout<<"Name:"<<custName<<endl;

cout<<"Account number:"<<accNo<<endl;

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

}

};

int main(){

BankAccount b;

BankAccount b1(101,"Adheesh",5000);

b.CreateAccount();

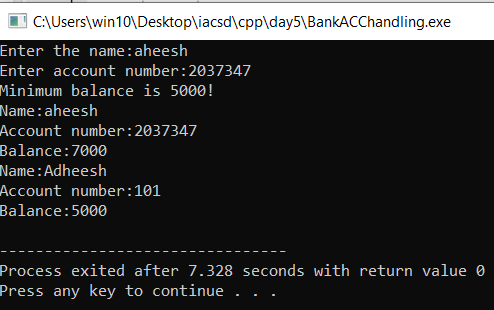
b.deposite(10000);

b.withdraw(3000);

b.display();

b1.display();

}



3. /\*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;

};

class Rect:public Shape{

private:

int l,b,A;

public:

void area(){

cout<<"Enter the l and b:"<<endl;

cin>>l>>b;

A=l\*b;

cout<<"Area="<<A<<endl;

}

};

class Circle:public Shape{

int r,A;

public:

void area(){

cout<<"Enter the r:"<<endl;

cin>>r;

A=3.14\*r\*r;

cout<<"Area="<<A<<endl;

}

};

class Square:public Shape{

int l,A;

public:

void area(){

cout<<"Enter the l:"<<endl;

cin>>l;

A=l\*l;

cout<<"Area="<<A<<endl;

}

};

int main(){

Circle c;

Rect r;

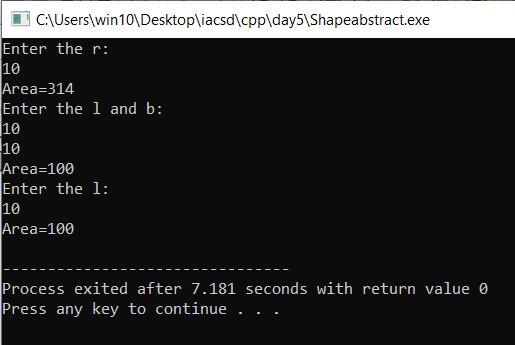
Square s;

c.area();

r.area();

s.area();

}



**Day6\_lab**

1. 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{

private:

int num;

int top=-1;

int stackarr[5];

public:

void push(int num){

try{

if(top>4){

throw 1;

}

else{

stackarr[++top]=num;

}

}

catch(int no){

cout<<"stack overflow exception"<<endl;

}

catch(...){

cout<<"exception occured"<<endl;

}

}

int pop(){

try{

if(top<0){

throw 1;

}

else{

num=stackarr[top--];

}

}

catch(int n){

cout<<"stack empty exception"<<endl;

}

catch(...){

cout<<"exception occured"<<endl;

}

return num;

}

void display(){

cout<<"\n\ntop="<<top;

if(top>-1){

cout<<"\n\nStack elements: ";

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

cout<<stackarr[i]<<" ";

}

}

}

};

int main(){

Stack s;

s.push(10);

s.push(20);

s.push(30);

s.push(40);

s.push(50);

s.push(60);

// s.push(90);

s.display();

//

int num2=s.pop();

int num3=s.pop();

int num4=s.pop();

int num5=s.pop();

//int num6=s.pop();

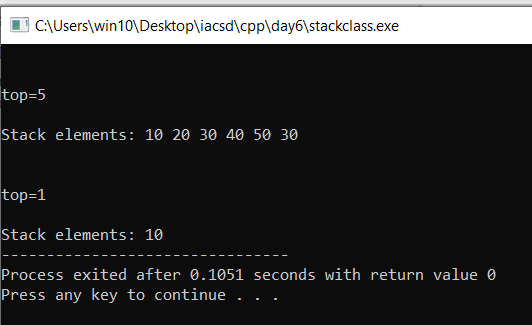
// int num7=s.pop();

//int num8=s.pop();

cout<<num5<<endl;

s.display();

}



!\_\_\_\_END\_\_\_\_!