# Tutorial 4

#include<iostream>

using namespace std;

int glo=6;

void sum(){

    int a;

    cout<<glo;

}

int main(){

    // int a=4;

    // int b=5;

    int glo=89;

    int a=4,b=7;

    float pi=3.145;

    char me='l';

    bool is =true;

    // cout<<"this is tutorial 4.Here the value of a is "<<a<<".the value of b is "<<b;

    // cout<<"\nthe value of pi is  "<<  pi;

    // cout<<"\n me is : "<<me;

    sum();

    cout<<glo<<is;   //first it will search in local function the value of glo then in main

    return 0;

}

# Tutorial 5;

# include<iostream>

using namespace std;

int main()

{

    int num1,num2;

    cout<<"enter the value of num1 : \n";

                            // insertion operator '<<'

    cin>>num1;              //'>>'is called extraction operator

    cout<<"enter the value of num2 : \n";

    cin>>num2;

    cout<<"the sum is "<< num1+num2;

    return 0;

}

# TUTORIAL 6

// there are two types of header files

// 1. system header files : it comes with compiler

#include<iostream>

// 2. user defined header files : it is written by the programmer

//#include"this.h"    this will produce error if is not present in the current directory

using namespace std;

int main(){

    int a=4,b=5;

    cout<<"Operator in c++"<<endl;

    cout<<"Following are operators in c++ "<<endl;

    // Arthmetic operators

    cout<<" value of a+b "<<a+b<<endl;

    cout<<" value of a-b "<<a-b<<endl;

    cout<<" value of a\*b "<<a\*b<<endl;

    cout<<" value of a%b "<<a%b<<endl;

    cout<<" value of a/b  "<<a/b<<endl;

    cout<<"value of a++ "<<a++<<endl;// first print 4 then value become 5

    cout<<"value of a-- "<<a--<<endl;// similarly

    cout<<"value of ++a "<<++a<<endl;// first increase then print

    // Assignment operators --> used to assign values to variables

    // int a=4,b=5;

    //Comparision operators

    cout<< "the value of a==b "<<(a==b)<<endl;

    cout<<"the value of a!=b  "<<(a!=b)<<endl;

    cout<<"the value of a<b  "<<(a<b)<<endl;

    //logical operators

    cout<< "the value of a==b && a<b  "<<((a==b) && (a<b))<<endl;

    cout<<"the value of a!=b or a==b  "<<((a!=b) || (a==b))<<endl;

    cout<<"the value of a<b  "<<!(a<b)<<endl;       // not operators

    return 0;

}

# Tutorial 7

#include<iostream>

using namespace std;

int c=45;

int main(){

    // \*\*\*\*\*\*\*\*build in data types

    //int a,b,c;

    //cout<<"enter tha value of a :"<<endl;

    //cin>>a;

    //cout<<"enter the value of b :"<<endl;

    //cin>>b;

    //c=a+b;

    //cout<<"the sum is c :"<<c;

    //cout<<"the globe c is  :"<<::c;       //scope resolution to print global value

    //\*\*\*\*\*\*Float,doubleand long double literals\*\*\*\*

    //float d=34.4f;         // if is not by default long will be given

    //long double e=34.4l;

    //cout<<"the size of 34.4f is"<<sizeof(34.4)<<endl;  //size of double

    //cout<<"the size of 34.4f is"<<sizeof(34.4f)<<endl;

    //cout<<"the size of 34.4F is"<<sizeof(34.4F)<<endl;

    //cout<<"the size of 34.4l is"<<sizeof(34.4l)<<endl;

    //cout<<"the size of 34.4L is"<<sizeof(34.4L)<<endl;

    //cout<<"the value d id "<<d<<endl<<"the value of e is "<<e;

     //\*\*\*\*\*\*reference variables\*\*\*\*

    // shubham-->shubh---->rohu----->dangerous coder

    float x=455;

    float & y=x;

    cout<<x<<endl;

    cout<<y<<endl;

     //\*\*\*\*\*\*type casting\*\*\*\*

    int a =45;

    float b=45.67;

    cout<<"the value of a is "<<(float)a<<endl;

    cout<<"the value of b is "<<(int)a<<endl;

    cout<<"the value of a+b "<<a+b<<endl;

    cout<<"the value of a+int(b) "<<a+int(b)<<endl;

    cout<<"the value of a+(int)b  "<<a +(int)b<<endl;

    return 0;

}

# Tutorial 8

//constants ,manipulators and operatot precedence

#include<iostream>

#include<iomanip>

using namespace std;

int main(){

    //int a=34;

    //cout<<"the value of a was"<<a;

    //a=45;

    //cout<<"the value of a is"<<a;

    //constants in c++

    //const int a=3;

    //cout<<"the value of a is"<<a<<endl;

    //a=5;         //we will get error as a is constant

    //cout<<"the value of a is"<<a<<endl;

//manipulator in c++  ex- endl;

    //int a=3,b=78,c=1233;

    // cout<<"the value of a is "<<a<<endl;

    // cout<<"the value of b is "<<b<<endl;

    // cout<<"the value of c is "<<c<<endl;

    // cout<<"the value of a is "<<setw(4)<<a<<endl;          // give space  of 4

    // cout<<"the value of b is "<<setw(4)<<b<<endl;

    // cout<<"the value of c is "<<setw(4)<<c<<endl;

    //\*\*\*\*\*\*\*operator precedence

    int a=3,b=4;

    int c=((((a\*5)+b)-45)+87);          //left to right

    cout<<c;

    return 0;

}

TUTORIAL 9

// if -else and control structures

//1.sequence structure

//2.selection structure

//3.loop structure

//these are basic control structure

//sequence structure

#include<iostream>

using namespace std;

int main(){

    //cout<<"this is tut 9";

    int age;

    cout<<"tell me your age "<<endl;

    cin>>age;

    //\*\*\*\*\*selection control if else...

    // if ((age<18)&&(age>0)){

    //     cout<<"you cannot come to my party"<<endl;

    // }

    // else if (age==18){

    //     cout<<"you are kid and not allowed"<<endl;

    // }

    // else if (age <=0){

    //     cout<<"you are not yet born"<<endl;

    // }

    // else{

    //     cout<<"you can come to the party"<<endl;

    // }

    // selection control structure : switch case statements

    switch (age)

    {

    case 18:

        /\* code \*/

        cout<<"you are 18"<<endl;

        break;

    case 22:

        /\* code \*/

        cout<<"you are 22"<<endl;

        break;                          // if break is not used then all condition after that will execute

    case 2:

        /\* code \*/

        cout<<"you are 2"<<endl;

        break;

    default:

    cout<<"no special case"<<endl;

        break;

    }

    cout<<"done with switch case";

    return 0;

}

TUTORIAL 10

#include<iostream>

using namespace std;

int main(){

    /\* for loop in c++

    there are 3 types of loops in c++

    1.For loop

    2.while loop

    3.do whilw loop

    \*/

   //for loop in c++

   // for (initialization; condition ;updation)

//    for (int i = 0; i < 4; i++)

//    {

//     /\* code \*/

//     cout<<i<<endl;

//     //i++;

//    }

/\*\*\*\*\*while loop in c++\*\*\*/

//syntax:

//printing 1 to 40 using while loop

// int i=1;

// while (i<=40){

//     cout<<i<<endl;

//     i++;

// }

int i=1;

do {

    cout<<i<<endl;

    i++;

}while(i<=40);

    return 0;

}

//question write multiplication of 6...HW

TUTORIAL 11

#include<iostream>

using namespace std;

int main(){

    // for (int i=0;i<4;i++)

    // {

    //     cout<<i<<endl;

    //     if(i==2){

    //         break;             // will break the loop when i==2

    //     }

    // }

    for(int i=0;i<40;i++)

    {

        if (i==2){

            continue;               // i==2 case only skip nothing else

        }

        cout<<i<<endl;

    }

    return 0;

}

TUTORIAL 12

#include<iostream>

using namespace std;

int main()

{

    //what is pointers??      data type whic holds the address of other data types

    int a=3;

    int\* b =&a;

    //&----> (address of) operator

    cout<<"the address of a is "<<&a<<endl;

    cout<<"the address of a is "<<b<<endl;

    //\*--->(value at) Derefernce operator

    cout<<"the value of address of b is "<<\*b<<endl;

    //pointer to pointer

    int\*\* c=&b;

    cout<<"the address of b is"<<&b<<endl;

    cout<<"the address of b is "<<c<<endl;

    cout<<"the value at address c is "<<\*c<<endl;

    cout<<"the value at address value\_at(value\_at(c))b is "<<\*\*c<<endl;

    return 0;

}

TUTORIAL 13

//array & pointers

#include<iostream>

using namespace std;

int main(){

    int marks[]={23,45,65,98};

    int mathmarks[4];

    mathmarks[0]=99;

    mathmarks[1]=89;

    mathmarks[2]=78;

    mathmarks[3]=99;

    cout<<"these are math marks"<<endl;

    cout<<mathmarks[0]<<endl;

    cout<<mathmarks[1]<<endl;

    cout<<mathmarks[2]<<endl;

    cout<<mathmarks[3]<<endl;

cout<<"these are   marks"<<endl;

    marks[2]=455;         //you can change the value of array

    // cout<<marks[0]<<endl;

    // cout<<marks[1]<<endl;

    // cout<<marks[2]<<endl;

    // cout<<marks[3]<<endl;

    // for(int i=0 ;i<4;i++){

    //     cout<<"the value of marks  "<<i<<"  is "<<marks[i]<<endl;

    // }

    //quiz :  do the same with while loop

    // int i=0;

    // while(i<4){

    //     cout<<"the value of marks "<<i<<"  is  "<<marks[i]<<endl;

    //     i++;

    // }

    //Pointers and arrays

    int\* p=marks;           // p is storing address of starting of array ( i.e marks[0])

    cout<<"the value of marks[0]/\*p is "<< \*(p)<<endl;        //script p will print value of mark[0],, \*(p+1) consecutive

    cout<<"the value of marks[1]/\*(p+1) is "<< \*(p+1)<<endl;

    cout<<"the value of marks[2]/\*(p+2) is "<< \*(p+2)<<endl;

    cout<<"the value of marks[3]/\*(p+3) is "<< \*(p+3)<<endl;

    return 0;

}

TUTORIAL 14

#include <iostream>

using namespace std;

//structure ,union and enums

typedef struct employee      // if data have different data types like ...int,char,float wecan do it in array

{

    /\* data \*/

    int eId;

    char favChar;

    float salary;

} ep;

// struct employee      // if data have different data types like ...int,char,float wecan do it in array

// {

//     /\* data \*/

//     int eId;

//     char favChar;

//     float salary;

// };

union money      // UNION     takes maximum size of memory and allot one at a time

{

    /\* data \*/

    int rice;

    char car;

    float pounds;

};

int main(){

    ep harry;

    ep shubham;

    ep ujjwal;

    // struct employee harry; if we are not using typedef  and instead of struct employee we can use ep

    // struct employee shubham;

    // struct employee ujjwal;

    union money m1;

    m1.rice=34;

    m1.car='r';

    cout<<"union results are  "<<m1.rice<<endl;

    ujjwal.salary=80000;

    harry.eId=1;

    harry.favChar='c';

    harry.salary=12000;

    cout<<"the value is  id is  "<<harry.eId<<endl;

    cout<<"the value is char  "<<harry.favChar<<endl;

    cout<<"the value is  salary  "<<harry.salary<<endl;

    cout<<"the value is  salary  "<<ujjwal.salary<<endl;

    cout<<"the value is  salary  "<<ujjwal.eId<<endl;

    return 0;

}

TUTORIAL 14

#include<iostream>

using namespace std;

int main(){

    enum meal {breakfast,lunch,dinner}; //enum

    meal m2=breakfast;

    cout<<m2;

    cout<<breakfast<<endl;

    cout<<lunch<<endl;

    cout<<dinner<<endl;

    return 0;

}

TUTORIAL 15

#include<iostream>

using namespace std;

// function prototype

//type function -name (arguments);

//int sum(int a,int b);          //Acceptable --->..

//int sum(int a,int);// not Acceptable ------>>>

int sum (int , int);           //Acceptable----->>>

//void g(void);        // no type no return

void g();         //also acceptable

int main(){

    int num1 ,num2;

    cout<<"enter first number"<<endl;

    cin>>num1;

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

    cin>>num2;

    // num1 and num2 are actual parameters

    cout<<"The sum is  "<<sum(num1,num2);

    g();

    return 0;

}

int sum (int a , int b){

    // a and b are formal parameter take value from actual

    int c =a+b;

    return c;

}

void g(){

    cout<<"\n good morning";

}

TUTORIAL 16

//call by value and references

#include<iostream>

using namespace std;

int sum(int a,int b){

    int c=a+b;

    return c;

}

//this will not swap a and b

void swap(int a,int b){

    int temp=a;

    a=b;

    b=temp;

}

//call by reference using pointers

void swapPointer(int\* a, int\* b){

    cout<<"the address of a "<<a<<endl;

    int temp=\*a;

    \*a=\*b;

    \*b=temp;

}

//call by reference using C++ reference variables

void swapReferenceVar(int  &a, int &b){            //& lecture 7

     //cout<<"the address of a "<<a<<endl;

    int temp=a;

    a=b;

    b=temp;

}

int main(){

    int x=4, y=5;

    //cout<<"The value of sum of 4 and 5 is "<<sum(a,b);

    cout<<"The value of x is  "<<x<<" and the value of y is "<<y<<endl;

    //swap(x,y); this will not swap a and b

    //swapPointer(&x,&y); // this will swap a and b

    swapReferenceVar(x,y);

    cout<<"The value of x is  "<<x<<" and the value of y is "<<y<<endl;

    return 0;

}

TUTORIAL 17

#include<iostream>

using namespace std;

inline int product(int a , int b){        //when in function work is less make it inline

    // static int c=0;  // Not recommended to use below lines with inline functions

                      // initialize/execute only one time

    // c=c+1;                   // next time this function is run ,the value of c will be retained

    return a\*b;

}

float moneyreceived(int currentmoney,float factor=1.04){

    return currentmoney\*factor;

}

int main(){

    int a ,b;

    cout<<" enter the value of a and b"<<endl;

    cin>>a>>b;

    // cout<<" the product of a and b is "<<product(a,b)<<endl;

    // cout<<" the product of a and b is "<<product(a,b)<<endl;

    // cout<<" the product of a and b is "<<product(a,b)<<endl;

    int money=100000;

    cout<<" if you have "<<money<<" Rs in Your bank account "<<moneyreceived(money)<<" Rs after 1 year "<<endl;

    cout<<" For  VIP :if you have "<<money<<" Rs in Your bank account "<<moneyreceived(money,1.1)<<" Rs after 1 year "; //DEFAULT ARG

        return 0;

}

TUTORIAL 18

#include<iostream>

using namespace std;

int factorial(int n){

    if (n<=1){

        return 1;

    }

      return n\*factorial(n-1);

}

int main(){

    //factorial of number

    int a;

    cout<<" enter a number ";

    cin>>a;

    cout<<"The factorial of factorial "<<factorial(a);

    return 0;

}

TUTORIAL 18 A

#include<iostream>

using namespace std;

int fib (int n){

    if (n<2){

        return 1;

    }

    return fib(n-1)+fib(n-2);

}

int main(){

    int a;

    cout<<"enter a number : "<<endl;

    cin>>a;

    cout<<"the term in fibonacci series at position "<<a<< " is "<< fib(a)<<endl;

    return 0;

}

TUTORIAL 21

#include<iostream>

using namespace std;

class employee

{

    private:

        int a,b,c;

    public:

        int d,e;

        void setData(int a,int b,int c); //Declaration

        void getData(){

        cout<<"The value of a is "<<a<<endl;

        cout<<"The value of b is "<<b<<endl;

        cout<<"The value of c is "<<c<<endl;

        cout<<"The value of d is "<<d<<endl;

        cout<<"The value of e is "<<e<<endl;

        }

};

void employee ::setData(int a1,int b1,int c1){

    a=a1;

    b=b1;

    c=c1;

}

int main(){

    employee harry;

    //harry.a=12; will not work as it is private

    harry.d=299;

    harry.e=99999;

    harry.setData(1,23,4);

    harry.getData();

    return 0;

}

TUTORIAL 22

    //OOPs -Classes and objects

    //class -->extension of structures

    //sturctures had limitations

    //  --> members are public

    //  --> no methods

    //classes -->  structures+ more

    //classes ---> can have methods and properties

    //classes ---> can make few members as private  and few as public

    //structures are in c++ are typedefed

    // you can declare objects along with th class declaration

    /\*class Employee {

        //class definition

    } harry ,rohan,lovish;\*/

    //harry.salary=9; makes no sense if salary is private

//Nesting of member function

#include<iostream>

#include<string>

using namespace std;

    class binary {

        private:    //by default class members are private

            void chk\_bin(void); // now only class methods can use chk\_bin

        string s;

    public:

        void read(void);

        //void chk\_bin(void);

        void ones\_compliment(void);

        void display(void);

    };

    void binary ::read (void){

        cout<<"enter a binary number ";

        cin>>s;

    }

    void binary ::chk\_bin(void){

        for(int i=0;i<s.length();i++){

            if (s.at(i)!='0' && s.at(i)!='1'){

                cout<<"incorrect"<<endl;

                exit(0);

            }

        }

    }

    void  binary :: ones\_compliment(void){

        chk\_bin();            // no need to write b.chk\_bin(); it will run for same object as of ones compliment

        for(int i=0;i<s.length();i++){

            if (s.at(i)=='1'){

                s.at(i)='0';

            }

            else

            {

                s.at(i)='1';

            }

        }

    }

    void binary :: display(void){

        for(int i =0;i<s.length();i++){

            cout<<s.at(i);

        }

    }

int main(){

    binary b;

    b.read();

    //b.chk\_bin();     //it can run in another way too        ,,,,will not run as it declared in private function

    b.ones\_compliment();

    b.display();

    return 0;

}

TUTORIAL 23

// memory allocation

#include <iostream>

using namespace std;

class Shop

{

    int itemId[100];    // by default private

    int itemPrice[100];

    int counter;

public:

    void initCounter(void) { counter = 0; }

    void setPrice(void);

    void displayPrice(void);

};

void Shop ::setPrice(void)

{

    cout << "Enter Id of your item no " << counter << endl;

    cin >> itemId[counter];

    cout << "Enter price of your item no " << counter << endl;

    cin >> itemPrice[counter];

    counter++;

}

void Shop ::displayPrice(void)

{

    for (int i = 0; i < counter; i++)

    {

        cout << " the price of item with Id " << itemId[i] << " is " << itemPrice[i] << endl;

    }

}

int main()

{

    Shop dukaan;

    dukaan.initCounter();

    dukaan.setPrice();

    dukaan.setPrice();

    dukaan.setPrice();

    dukaan.displayPrice();

    return 0;

}

TUTORIAL 24

#include<iostream>

using namespace std;

class Employee{

    public:

    int id;

    static int count;            //by default initialize by 0 //class variable

        void setData(void){

            cout<<"Enter the id "<<endl;

            cin>>id;

            count++;

        }

        void getData(void){

            cout<<"the id of this employee id  "<<id<<"and this is employee number :"<<count<<endl;

        }

        static void getCount(void){          //only for static variable

            cout<<"the value of count is "<<count<<endl;

        }

};

int Employee ::count; //count=100 is also fine here

                    //count is static data member of class employee

int main(){

    Employee harry,raj,luv;

    harry.setData();

    harry.getData();

    Employee::getCount();

    luv.setData();

    luv.getData();

    Employee::getCount();

    raj.setData();

    raj.getData();

    Employee::getCount();

    return 0;

}

TUTORIAL 25

#include<iostream>

using namespace std;

class Employee {

    int id;

    int salary;

    public :

        void setId(void){

            salary=122;

            cout<<"Enter the Id of employee "<<endl;

            cin>>id;

        }

        void getId(void){

            cout<<"the id of this employee is "<<id<<endl;

        }

};

int main(){

    // Employee harry,luv,rohan,shruti;

    // harry.setId();

    // harry.getId();

    Employee fb[4];         //array of object

    for(int i=0;i<4;i++){

        fb[i].setId();

        fb[i].getId();

    }

    return 0;

}

TUTORIAL 26

#include<iostream>

using namespace std;

class complex {

    int a,b;

    public:

        void setNumber(int n1 ,int n2){

            a=n1;

            b=n2;

        }

        friend complex Sumcomplex( complex o1, complex o2);  // non member func sumcomplex to give access to private data of complex

        void printNumber(){

        cout<<"your number is "<<a<<" + "<<b<<"i"<<endl;

        }

};

complex Sumcomplex( complex o1 ,complex o2){         // friend function

    complex o3;

    o3.setNumber( o1.a+o2.a ,o1.b+o2.b);

    return o3;

}

int main(){

    complex c1,c2,sum;

    c1.setNumber(1,4);

    c2.setNumber(5,8);

    c1.printNumber();

    c2.printNumber();

    sum=Sumcomplex( c1 , c2);

    sum.printNumber();

    return 0;

}

/\*properties of friend functions

1. Not in the scope of the class

2. since it is not in the scope of the class ,it cannot be called from the object of that class . c1.sumcomplex ()==invalid

3. can be invoked without the help of any object

4. usually contaion the object as arguments

5. can be delared public or private section of the class

6. it cannot access the memebers directly by their names and need object\_name.member\_name to access any member

\*/

TUTORIAL 27

//friend class

#include <iostream>

using namespace std;

// forward declaration

class complex;

class calculator

{

public:

    int add(int a, int b)

    {

        return (a + b);

    }

    int sumRealComplex(complex, complex); // we cannot write complex o1 ,complex 02 as it is not declared yet

    int sumImagineComlpex( complex , complex );

};

class complex

{

    int a, b;

    // individually declaring as friends (will become tough as function will increase)

    // friend int calculator ::sumRealComplex(complex o1, complex o2);

    // friend int calculator ::sumImagineComlpex (complex o1 , complex o2);

    //Alter : declaring entire class as a friend

    friend class calculator;

public:

    void setNumber(int n1, int n2)

    {

        a = n1;

        b = n2;

    }

    void printNumber()

    {

        cout << "your number is " << a << " + " << b << "i" << endl;

    }

};

int calculator ::sumRealComplex(complex o1, complex o2)

{

    return (o1.a + o2.a);

}

int calculator ::sumImagineComlpex(complex o1 , complex o2){

    return (o1.b + o2.b);

}

int main()

{

    complex o1, o2;

    o1.setNumber(1, 4);

    o2.setNumber(5, 7);

    calculator calc;

    int res = calc.sumRealComplex(o1, o2);

    cout << "The sum of real part of o1 and o2 is  " << res << endl;

    int res2 = calc.sumImagineComlpex(o1 , o2);

    cout << "The sum of imaginary part of o1 and o2 is  " << res2 << endl;

    return 0;

}

TUTORIAL 28

#include<iostream>

using namespace std;

class Y;

class X{

    int data;

    public:

        void setValue(int value){

            data=value;

        }

    friend void add ( X , Y);

};

class Y{

    int num;

    public:

        void setValue(int value){

            num=value;

        }

    friend void add ( X , Y);

};

void add ( X o1 ,Y o2 ){

    cout<<" summing data of X and Y objects gives "<<o1.data+o2.num;

 }

int main(){

    X a;

    a.setValue(3);

    Y b;

    b.setValue(5);

    add(a , b);

    return 0;

}

TUTORIAL 28 A

#include<iostream>

using namespace std;

class c2;

class c1{

    int val;

    public:

        void inData(int a){

            val = a;

        }

        void display(void){

            cout<< val <<endl;

        }

        friend void exchange( c1 & , c2 &);

};

class c2{

    int val2;

    public:

        void inData(int a){

            val2 = a;

        }

        void display(void){

            cout<< val2 <<endl;

        }

        friend void exchange( c1 &, c2 &);

};

//call bh reference

void exchange(c1 & x , c2 & y){

    int tmp=x.val;

    x.val=y.val2;

    y.val2=tmp;

}

int main(){

    c1 oc1;

    c2 oc2;

    oc1.inData(34);

    oc2.inData(67);

    exchange( oc1 , oc2);

    cout<<"The value of c1 after exchange becomes : ";

    oc1.display();

    cout<<"The value of c2 after exchange becomes : ";

    oc2.display();

    return 0;

}

TUTORIAL 29

//setdata without passing function and all

#include<iostream>

using namespace std;

class Complex {

    int a,b;

    public:

    //creating a constructor ...,,

    //comstructor is a special member function with same name as of the  class.It is automatically invoked whenever object is created

    //it is used to initialize the objects of its class

    Complex(void);          //Constructor declaration          retrun type is not impotant but imp in other member function

    void printdata()

    {

        cout<< " Your number is "<<a <<"  +  "<<b<<" i"<<endl;

    }

};

Complex :: Complex(void)     //default constructor as it takes no parameter

{

    a=10;

    b=0;

}

int main(){

    Complex c1,c2,c3;

    c1.printdata();

    c2.printdata();

    c3.printdata();

    return 0;

}

// properties of constructors

/\*

1. It should be declared in the puclic section of the class

2. They are automatically invoked whenever the object is created

3. Donot have return type and they cannot return value

4. It can have default arguments

5. We cannot refer to their address

\*/

TUTORIAL 30

#include <iostream>

using namespace std;

class Complex

{

    int a, b;

public:

    Complex(int, int);

    void printnumber()

    {

        cout << "your number is " << a << " + " << b << "i" << endl;

    }

};

Complex ::Complex(int x, int y) // parametrized constructor takes value here it is taking 2 parameter

{

    a = x;

    b = y;

}

int main()

{

    // Implicit call

    Complex a(4, 6);

    // explicit call

    Complex b = Complex(5, 7);

    a.printnumber();

    b.printnumber();

    return 0;

}

TUTORIAL 30 A

#include<iostream>

using namespace std;

class Point {

    int x,y;

    public :

        Point(int a , int b){

            x=a;

            y=b;

        }

        void displayPoint()

        {

            cout << " the point is ("<<x<<" , "<<y<<")"<<endl;

        }

};

//Create a function (hint: Make it a friend function) which takes 2 point objects and computes the distance between those 2 points

int main(){

    Point p(1,1);

    p.displayPoint();

    Point q(4,6);

    q.displayPoint();

    return 0;

}

ASSIGN 30

#include<iostream>

#include<math.h>

using namespace std;

class Point {

    int x,y;

    public :

        Point(int a , int b){

            x=a;

            y=b;

        }

        friend int distance(Point , Point);

        void displayPoint()

        {

            cout << " the point is ("<<x<<" , "<<y<<")"<<endl;

        }

};

int distance( Point t , Point s){

    int d;

    d=(t.x-s.x);

    return d;

}

//Create a function (hint: Make it a friend function) which takes 2 point objects and computes the distance between those 2 points

int main(){

    int sum;

    Point p(1,1);

    p.displayPoint();

    Point q(4,6);

    q.displayPoint();

    sum =distance(p,q);

    cout<<"the distance is "<< sum;

    return 0;

}

TUTORIAL 31

// CONSTRUCTOR OVERLOADING

#include<iostream>

using namespace std;

class complex {

    int a,b;

    public:

        complex (int x , int y){

            a=x;

            b=y;

        }

        complex (int x){

            a=x;

            b=0;

        }

        complex(void){

            a=0;

            b=0;

        }

        void printnumber(){

            cout<< " tour number is "<< a <<" + "<< b <<"i"<<endl;

        }

};

int main(){

    complex c1(4,6);

    c1.printnumber();

    complex c2(9);

    c2.printnumber();

    complex c3;

    c3.printnumber();

    return 0;

}

TUTORIAL 32

#include<iostream>

using namespace std;

class simple {

    int data1,data2;

    public :

    simple (int a , int b=9)       //default value of b

    {

        data1=a;

        data2=b;

    }

    void printnumber(){

            cout<< " tour data is "<< data1 <<" and "<< data2 <<endl;

        }

};

int main(){

    simple s(1,4),s1(3);

    s.printnumber();

    s1.printnumber();

    return 0;

}