Abstract class:

#include<iostream>

#include<iostream>

#include<string>

using namespace std;

class animal

{

public:

virtual void eat()=0;

};

class dog:public animal

{

public:

void eat()

{

cout<<"dog eats meat";

}

};

class cat:public animal

{

public:

void eat()

{

cout<<"cat eats fish"<<endl;

}

};

int main()

{

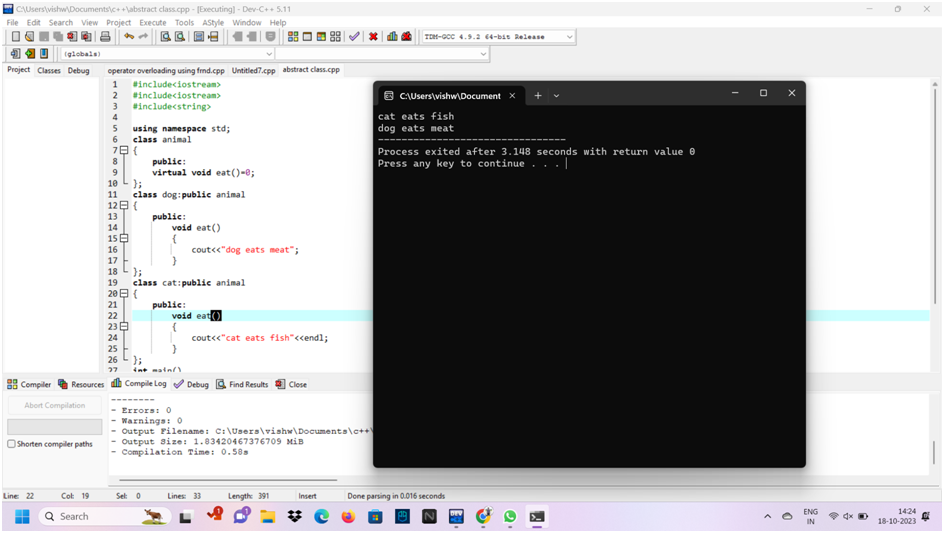
cat c1;

dog d1;

c1.eat();

d1.eat();

}



2.operator overloading using friend:

#include<iostream>

using namespace std;

class B;

class A{

public:

int a;

A(){

cout<<"enter the number: ";

cin>>a;

}

friend void operator+(A,B);

friend void operator-(A, B);

friend void operator\*(A,B);

friend void operator/(A, B);

};

class B{

public:

float b;

B(){

cout<<"enter the number: ";

cin>>b;

}

friend void operator+(A,B);

friend void operator-(A, B);

friend void operator\*(A,B);

friend void operator/(A, B);

};

void operator+(A a1,B b1){

cout<<"Addition: ";

cout<<a1.a+b1.b<<endl;

}

void operator-(A a1,B b1){

cout<<"Subtraction: ";

cout<<a1.a-b1.b<<endl;

}

void operator\*(A a1,B b1){

cout<<"multiplication: ";

cout<<a1.a\*b1.b<<endl;

}

void operator/(A a1,B b1){

cout<<"division: ";

cout<<a1.a/b1.b<<endl;

}

int main(){

A A1;

B B1;

A1+B1;

A1-B1;

A1\*B1;

A1/B1;

}

