

**JAMIA HAMDARD**

**OOP’s LAB FILE**

**MADE BY:** Faisal Mujeeb

**Enroll No:** 2020-310-057

**COURSE:** BTECH CSE

**SECTION:** A

**SUBMITTED TO:** ANAM MA’AM

**Index**

|  |  |  |
| --- | --- | --- |
| S.No. | Topic | Sign. |
| 1. | Program to create class Bank, having member id, bank balance and amount to be withdrawn. Take input from user the amount to be withdrawn and show remaining balance. |  |
| 2. | Write a program to use public and private data members. |  |
| 3. | Write a class with having static function and data member. |  |
| 4. | Write a class having default constructor, parametrized constructor and destructor in it. |  |
| 5. | Write a program which is showing single level inheritance. |  |
| 6. | Write a program which is showing multi level inheritance. |  |
| 7. | Write a program which is showing multiple inheritance. |  |
| 8. | Write a program which is showing hierarchical inheritance. |  |
| 9. | Write a program which is showing hybrid inheritance. |  |
| 10. | Write a program showing inheritance with constructors and destructors |  |
| 11. | Write a program having copy constructor in it. |  |
| 12. | Write a program of inheritance of classes using virtual keyword |  |
| 13. | Write a program to implement function overloading which having arguments of different datatype. |  |
| 14. | Write a program to implement function overloading which having different number of arguments. |  |
| 15. | Write a program to implement function overloading which having different order of arguments. |  |
| 16. | Write a program to implement operator overloading of increment unary operator. |  |
| 17. | Write a program to implement operator overloading of decrement unary operator. |  |
| 18. | Write a program to implement function overriding. |  |
| 19. | Write a program to implement abstract class using access specifiers. |  |
| 20. | Write a program to implement abstract class using access specifiers. |  |
| 21. | Write a program of class Date and then show day, month and year using unary operator |  |

**Problem 1: Program to create class Bank, having member id, bank balance and amount to be withdrawn. Take input from user the amount to be withdrawn and show remaining balance.**

**Code:**

#include <iostream>

using namespace std;

class Bank

{

    int deposit;

    int withdrawl;

    int balance;

    char name;

    int acn;

    int x,y;

public:

    Bank()

{

    std::cout<<"Default Constructor Called";

}

    Bank(int,int )

{

    x=10;

    y=20;

    std::cout<<"Parameterized constructor called";

    std::cout<<endl<<x<<endl<<y;

}

   static void pinn()

{

        static int pin;

        std::cout<<"Enter your pin:";

        std::cin>>pin;

        std::cout<<endl;

        if(pin==123)

        {

                std::cout<<"Correct pin";

                std::cout<<endl;

        }

        else

        {

                std::cout<<"Incorrect pin";

                std::cout<<endl;

        }

}

    void info()

{

    std::cout<<"Enter your name: ";

    std::cin>>name;

    std::cout<<endl;

    std::cout<<"Enter the account number: ";

    std::cin>>acn;

    std::cout<<endl;

}

    void deposit1()

{

    std::cout<<"Eenter the amount you want to deposit: ";

    std::cin>>deposit;

    std::cout<<endl;

}

    void withdrawl1()

{

    std::cout<<"Enter the amount you want to withdrawl: ";

    std::cin>>withdrawl;

    std::cout<<endl;

}

    void display()

{

    std::cout<<"Your details are as follows...."<<endl;

    std::cout<<"Name:"<<name<<endl;

    std::cout<<"Account Number: "<<acn<<endl;

    std::cout<<"Original amount in account: "<<deposit<<endl;

    std::cout<<"Withdrawled amount: "<<withdrawl<<endl;

    balance = deposit - withdrawl;

    std::cout<<"Balance: "<<balance;

}

    ~Bank()

{

    std::cout<<"Default Disstructor Called";

}

};

 int main()

{

    Bank b1;

    Bank::pinn();

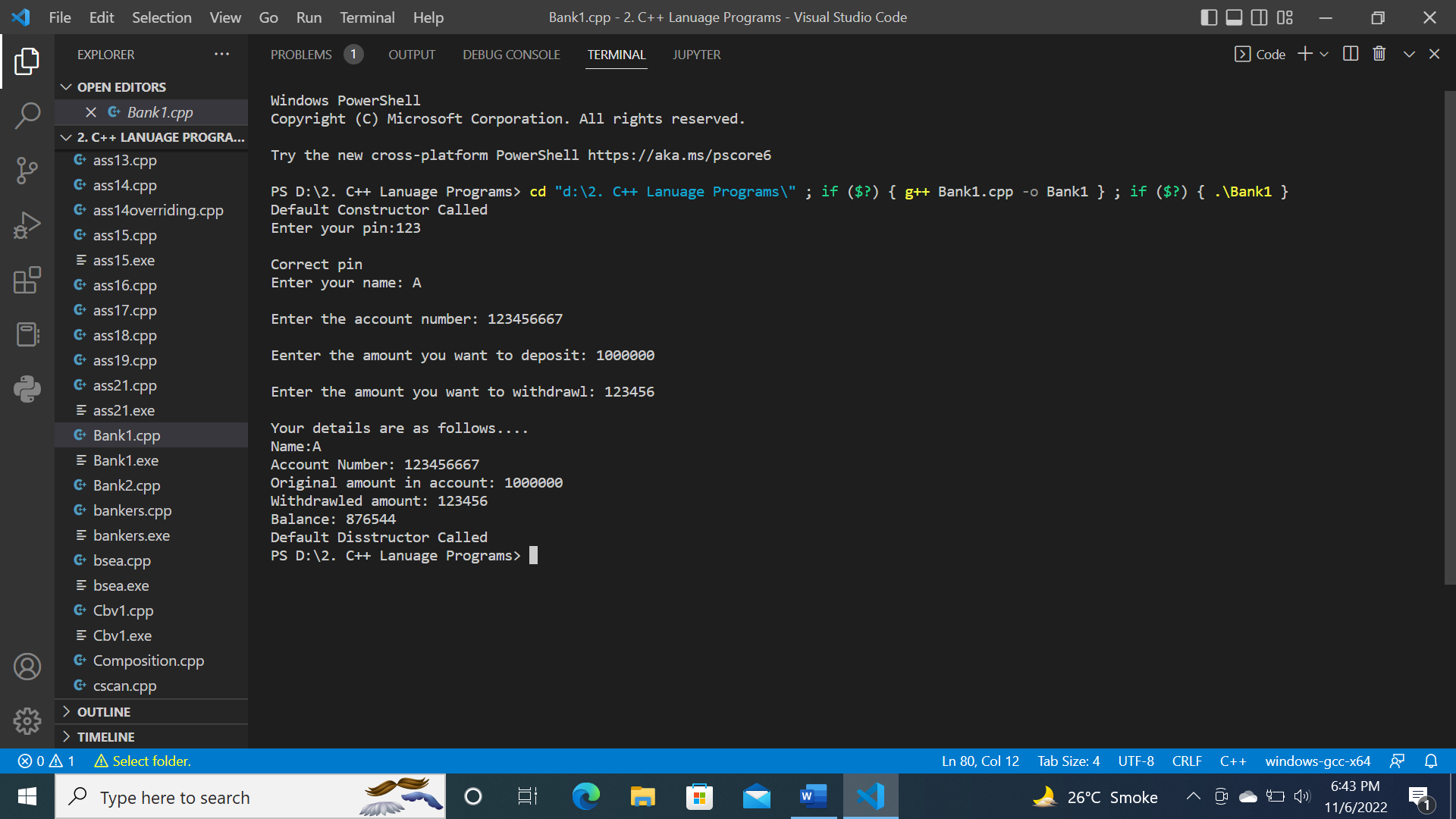
    b1.info();

    b1.deposit1();

    b1.withdrawl1();

    b1.display();

}



**Program 2: Write a program to use public and private data members.**

**Code:**

#include<iostream>

using namespace std;

class calculate

{

    private:

    int num1, num2, total;

    public:

    calculate(int, int);

    void sum();

};

calculate::calculate(int x,int y)

{

    num1 = x;

    num2 = y;

    total = 0;

}

void calculate::sum()

{

    total = num1 + num2;

    cout<<"sum is: "<<total<<endl;

}

int main()

{

    int var1, var2;

    cout<<"enter value 1: "<<endl;

    cin>>var1;

    cout<<"enter value 2: "<<endl;

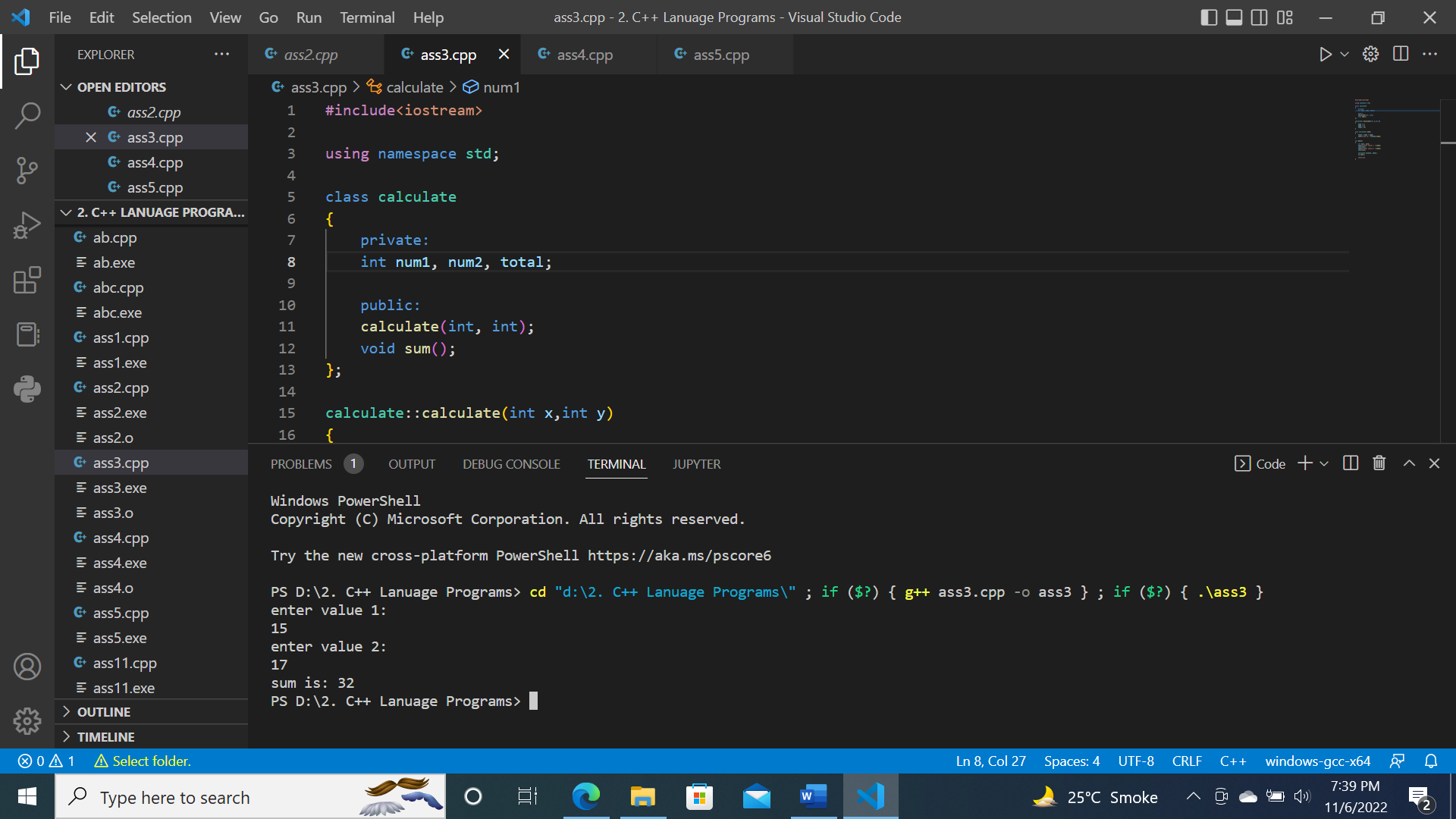
    cin>>var2;

    calculate c1(var1, var2);

    c1.sum();

    return 0;

}



**Program 3: Write a class with having static function and data member.**

**Code:**

#include <iostream>

#include <string>

using namespace std;

string name;

int account\_number;

int balance = 100000;

int withdrawn;

int avai\_bal;

int acc\_pin;

class banking

{

public:

    void get\_name()

    {

        cout << "Enter your name: " << endl;

        cin >> name;

    }

    void get\_acc()

    {

        cout<<""<<endl;

        cout << "Enter your account number: " << endl;

        cin >> account\_number;

        switch (account\_number)

        {

        case 12345:

            cout << "please proceed\n" << endl;

            break;

        default:

            cout << "wrong account number!" << endl;

            break;

        }

    }

    static void pin()

    {

        cout << "Enter your pin: " << endl;

        cin >> acc\_pin;

        switch (acc\_pin)

        {

        case 123:

            cout << "Correct Pin!\n" << endl;

            cout << "Balance in your account is : " << balance << endl;

            cout<<""<<endl;

            cout << "Enter amount to withdraw: " << endl;

            cin >> withdrawn;

            cout << " " << endl;

            avai\_bal = balance - withdrawn;

            if (avai\_bal < 0)

            {

                cout << "Sorry! not enough funds! " << endl;

            }

            else

            {

                cout << "The amount left in your account is : RS" << avai\_bal << endl;

            }

            break;

        default:

            cout << "Incorrect Pin!" << endl;

            break;

        }

    }

};

int main()

{

    banking b1;

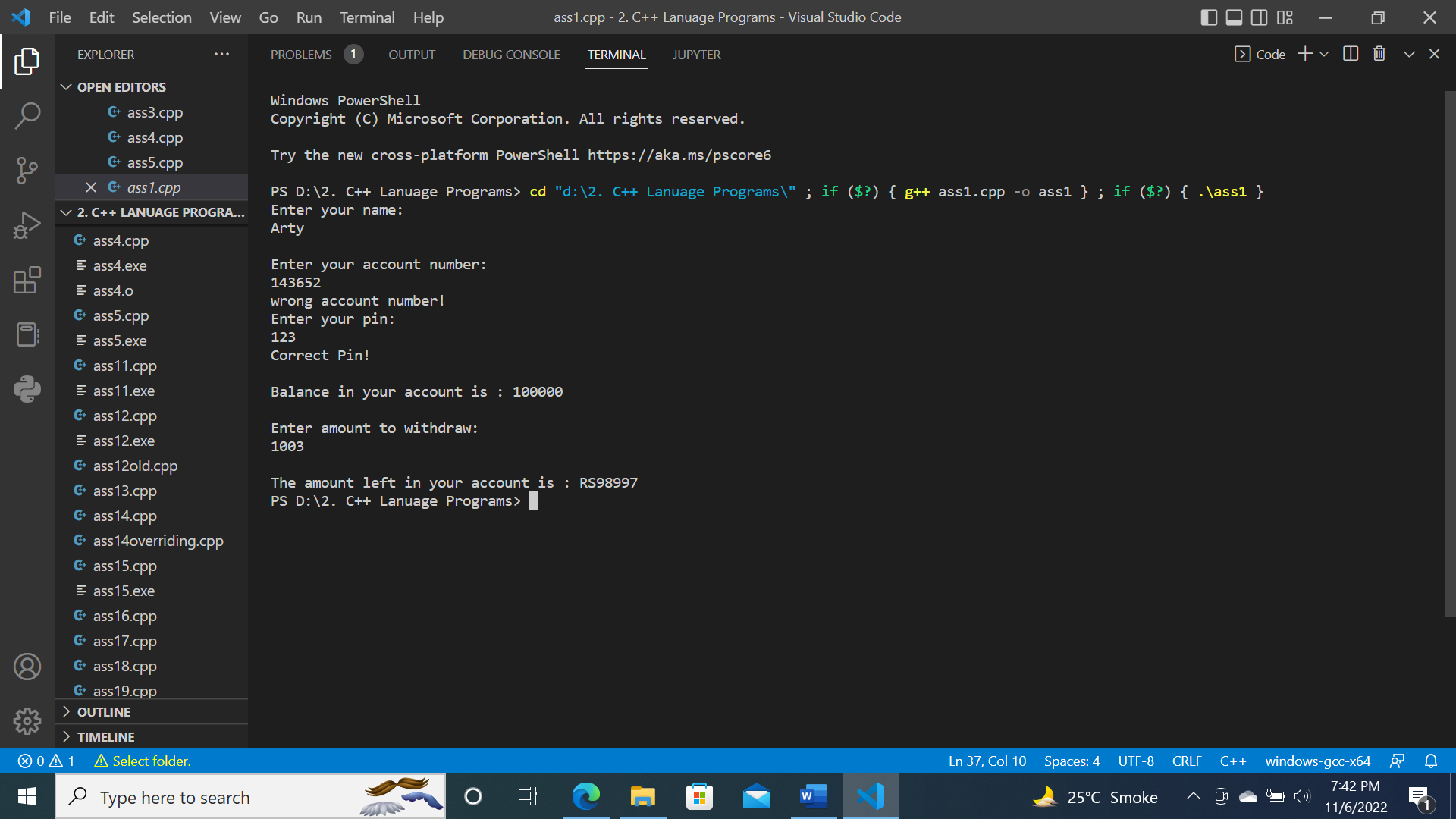
    b1.get\_name();

    b1.get\_acc();

    b1.pin();

    return 0;

}



**Program 4: Write a class having default constructor, parametrized constructor and destructor in it.**

**Code:**

#include <iostream>

#include <string>

using namespace std;

string name;

int account\_number;

int balance = 100000;

int withdrawn;

int avai\_bal;

int acc\_pin;

int x,y;

class banking

{

public:

    banking()

    {

        cout<<"default constructor invoked!"<<endl;

    }

    ~banking()

    {

        cout<<"default destructor invoked!"<<endl;

    }

    //parameterised constructor

    banking(int x,int y)

    {

        x = 4;

        y = 5;

        cout<<"parameterised constructor invoked!"<<endl;

        cout<<"x is : "<<x<<" y is : "<<y<<endl;

    }

    void get\_name()

    {

        cout << "Enter your name: " << endl;

        cin >> name;

    }

    void get\_acc()

    {

        cout<<""<<endl;

        cout << "Enter your account number: " << endl;

        cin >> account\_number;

        switch (account\_number)

        {

        case 12345:

            cout << "please proceed\n" << endl;

            break;

        default:

            cout << "wrong account number!" << endl;

            break;

        }

    }

    static void pin()

    {

        cout << "Enter your pin: " << endl;

        cin >> acc\_pin;

        switch (acc\_pin)

        {

        case 123:

            cout << "Correct Pin!\n" << endl;

            cout << "Balance in your account is : " << balance << endl;

            cout<<""<<endl;

            cout << "Enter amount to withdraw: " << endl;

            cin >> withdrawn;

            cout << " " << endl;

            avai\_bal = balance - withdrawn;

            if (avai\_bal < 0)

            {

                cout << "Sorry! not enough funds! " << endl;

            }

            else

            {

                cout << "The amount left in your account is : RS" << avai\_bal << endl;

            }

            break;

        default:

            cout << "Incorrect Pin!" << endl;

            break;

        }

    }

};

int main()

{

    banking b1;

    banking b2(x,y);

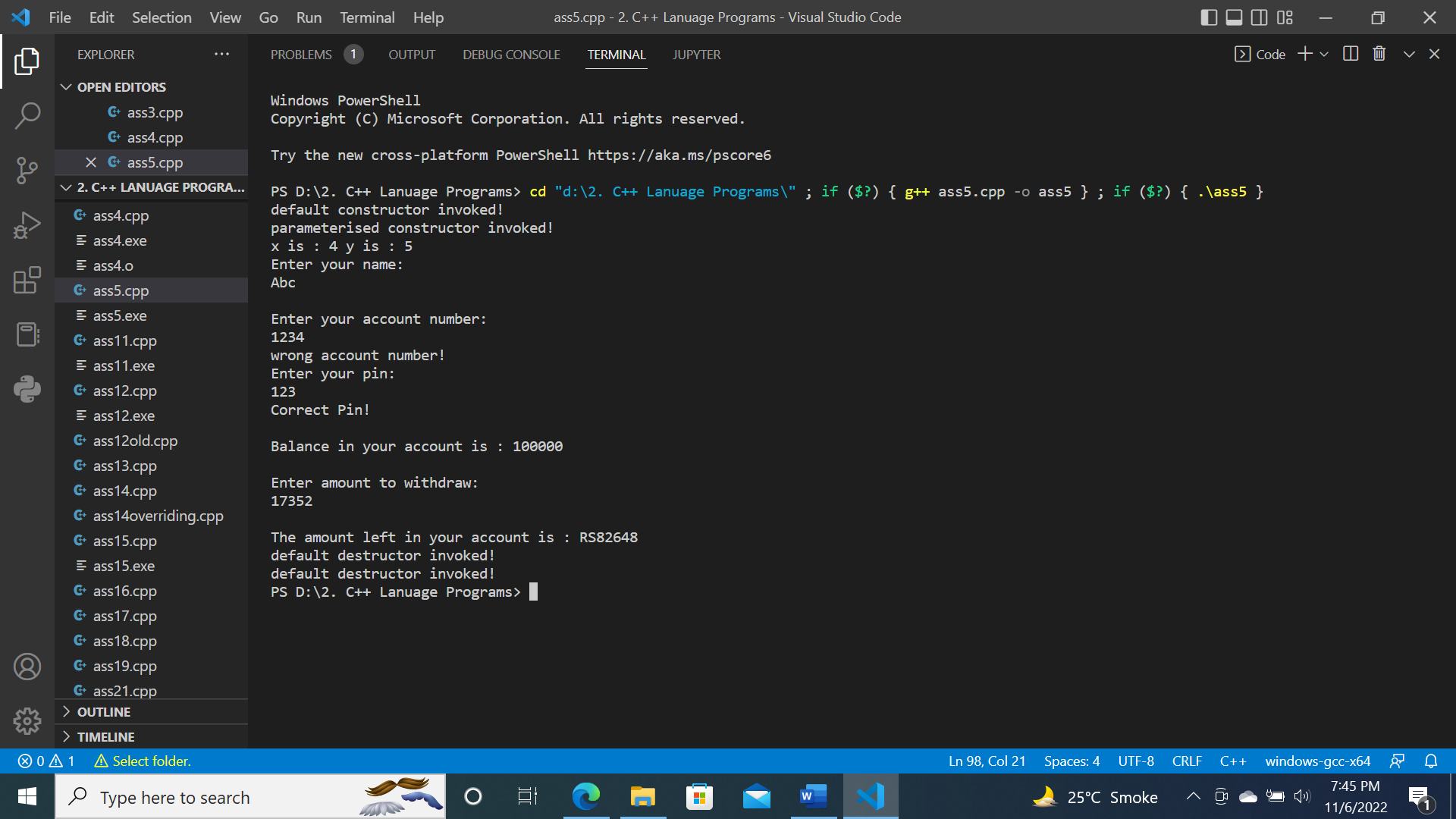
    b1.get\_name();

    b1.get\_acc();

    b1.pin();

    return 0;

}



**Program 5: Write a program which is showing single level inheritance.**

**Code:**

#include<iostream>

using namespace std;

int deposit;

    int withdrawl;

    int balance;

    char name;

    int acn;

    int x,y;

class Bank

{

public:

    void info()

{

    std::cout<<"Enter your name: ";

    std::cin>>name;

    std::cout<<endl;

    std::cout<<"Enter the account number: ";

    std::cin>>acn;

    std::cout<<endl;

}

    void deposit1()

{

    std::cout<<"Eenter the amount you want to deposit: ";

    std::cin>>deposit;

    std::cout<<endl;

}

    void withdrawl1()

{

    std::cout<<"Enter the amount you want to withdrawl: ";

    std::cin>>withdrawl;

    std::cout<<endl;

}

};

class Details: public Bank

{

public:

    void display()

{

    std::cout<<"Your details are as follows...."<<endl;

    std::cout<<"Name:"<<name<<endl;

    std::cout<<"Account Number: "<<acn<<endl;

    std::cout<<"Original amount in account: "<<deposit<<endl;

    std::cout<<"Withdrawled amount: "<<withdrawl<<endl;

    balance = deposit - withdrawl;

    std::cout<<"Balance: "<<balance;

}

};

 int main()

{

    Details d1;

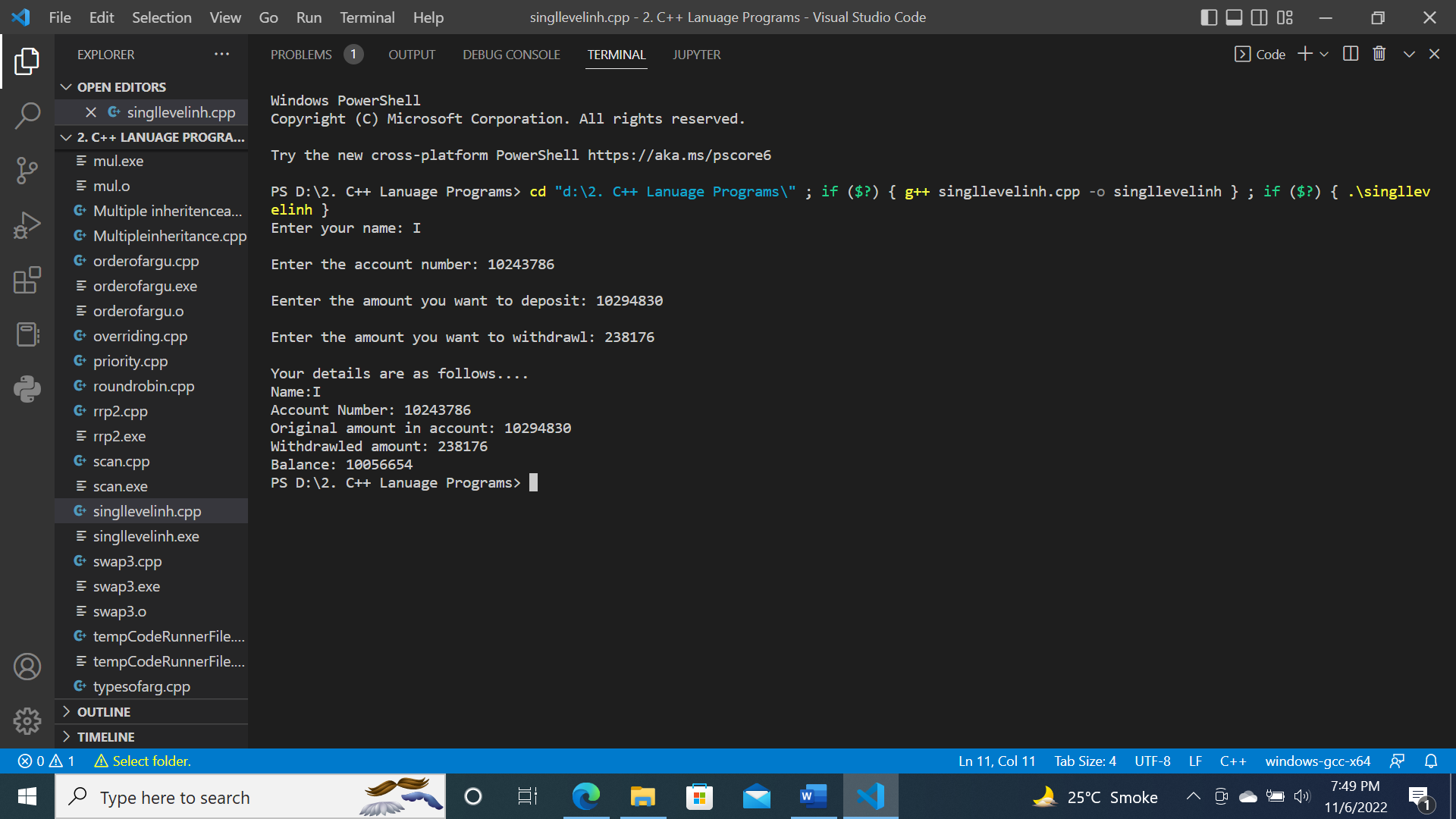
    d1.info();

    d1.deposit1();

    d1.withdrawl1();

    d1.display();

}



**Program 6: Write a program which is showing multi-level inheritance.**

**Code:**

#include <iostream>

using namespace std;

class A

{

    public:

      void display()

      {

          cout<<"Base class content.";

      }

};

class B:public A {};

class C:public B {};

int main()

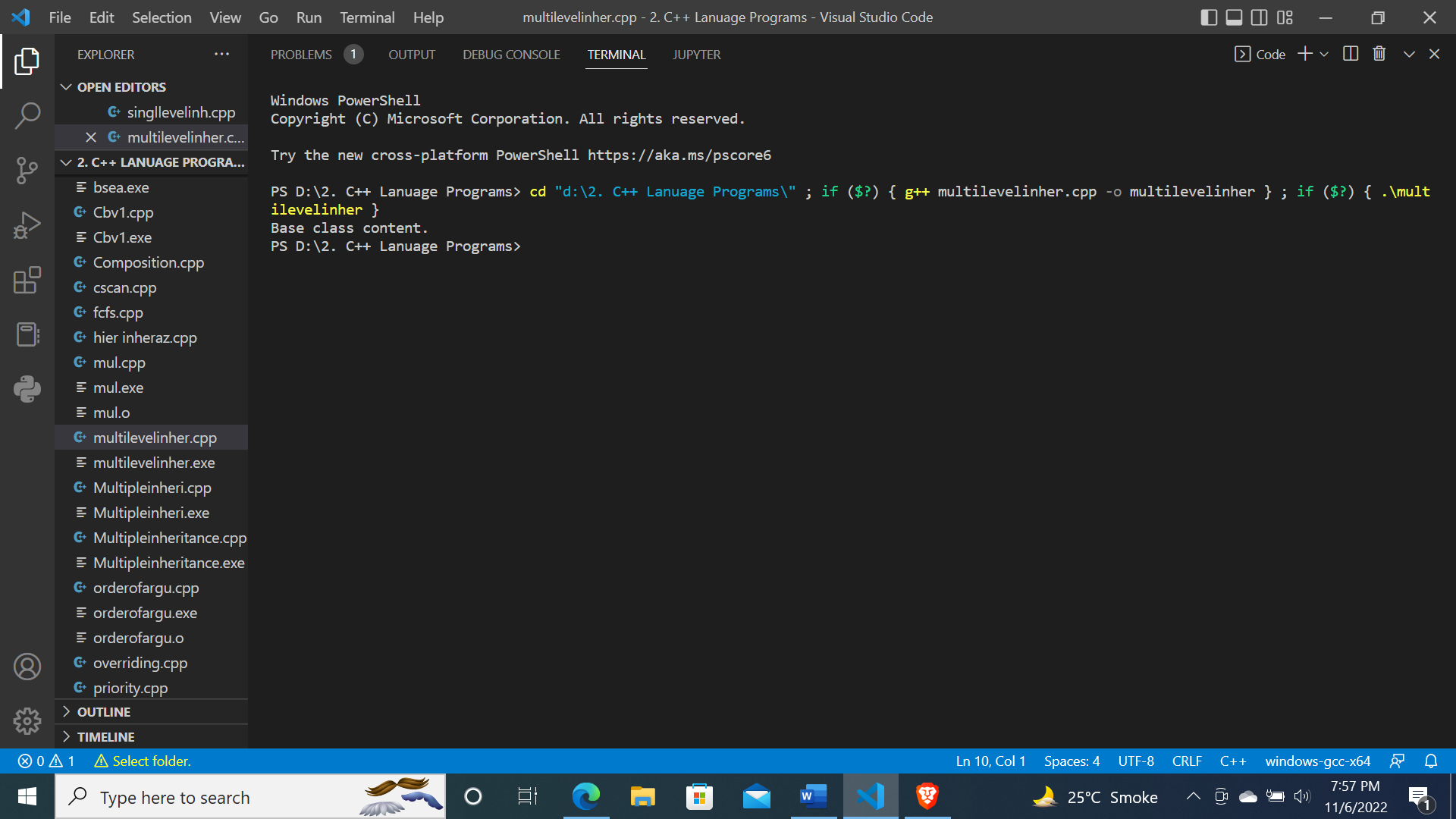
{

    C obj;

    obj.display();

    return 0;

}



**Program 7: Write a program which is showing multiple inheritance.**

**Code:**

#include<iostream>

using namespace std;

class Base1

{

  protected:

    int ivar1;

  public:

    void show\_1()

    {

        std::cout<<ivar1<<endl;

    }

};

class Base2

{

  protected:

    int ivar2;

  public:

    void show\_2()

    {

        std::cout<<ivar2<<endl;

    }

};

class Derived: public Base1, public Base2

{

  public:

    void set(int x, int y)

    {

        ivar1=x;

        ivar2=y;

    }

};

int main()

{

    Derived d;

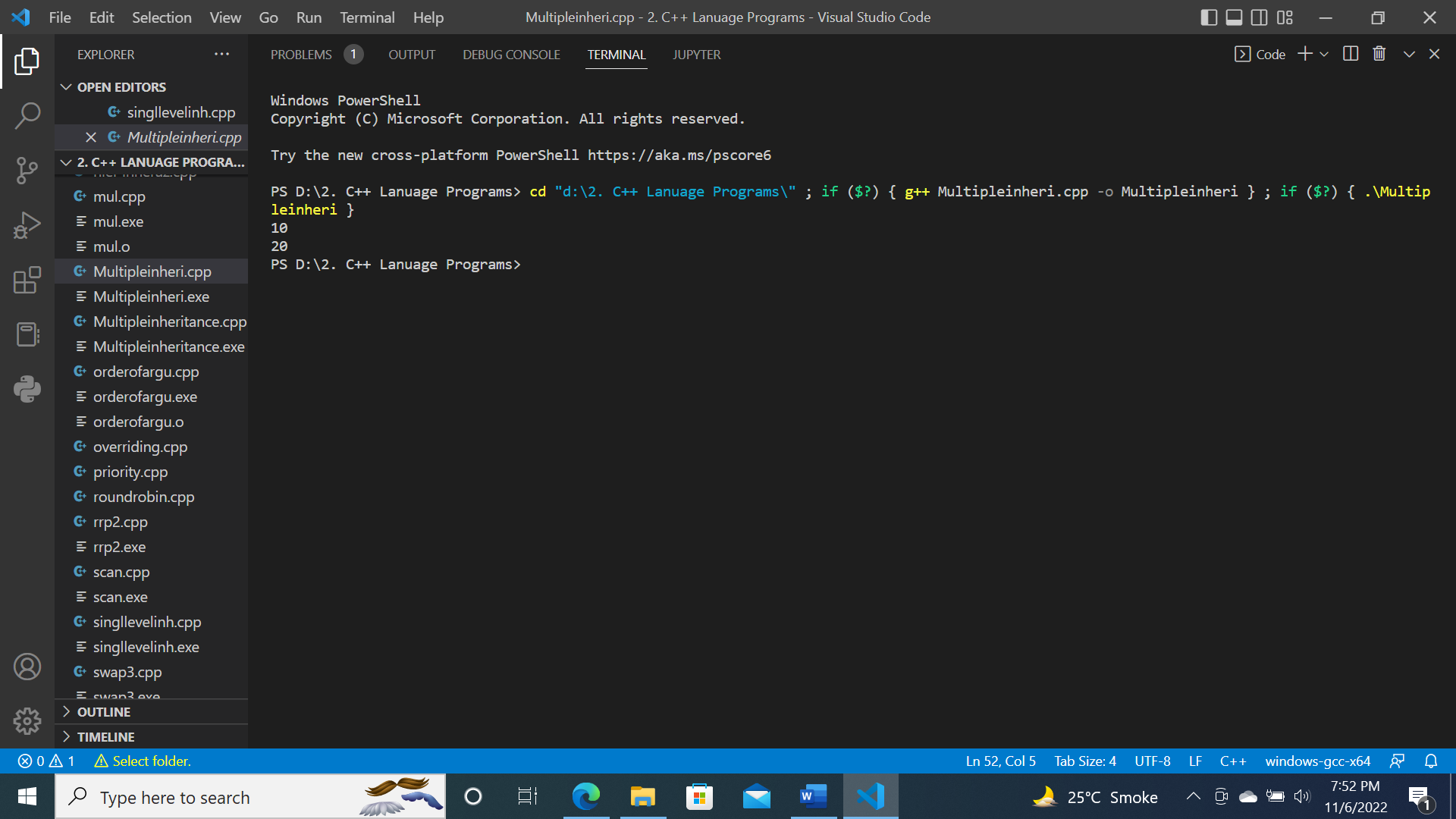
    d.set(10,20);

    d.show\_1();

    d.show\_2();

    return 0;

}



**Program 8: Write a program which is showing hierarchical inheritance.**

**Code:**

#include <iostream>

using namespace std;

class Animal

{

   public:

    void info() {

        cout << "I am an animal." << endl;

    }

};

class Dog : public Animal

{

   public:

    void bark() {

        cout << "I am a Dog." << endl;

    }

};

class Cat : public Animal

{

   public:

    void meow() {

        cout << "I am a Cat." << endl;

    }

};

int main()

{

    Dog dog1;

    cout << "Dog Class:" << endl;

    dog1.info();

    dog1.bark();

    Cat cat1;

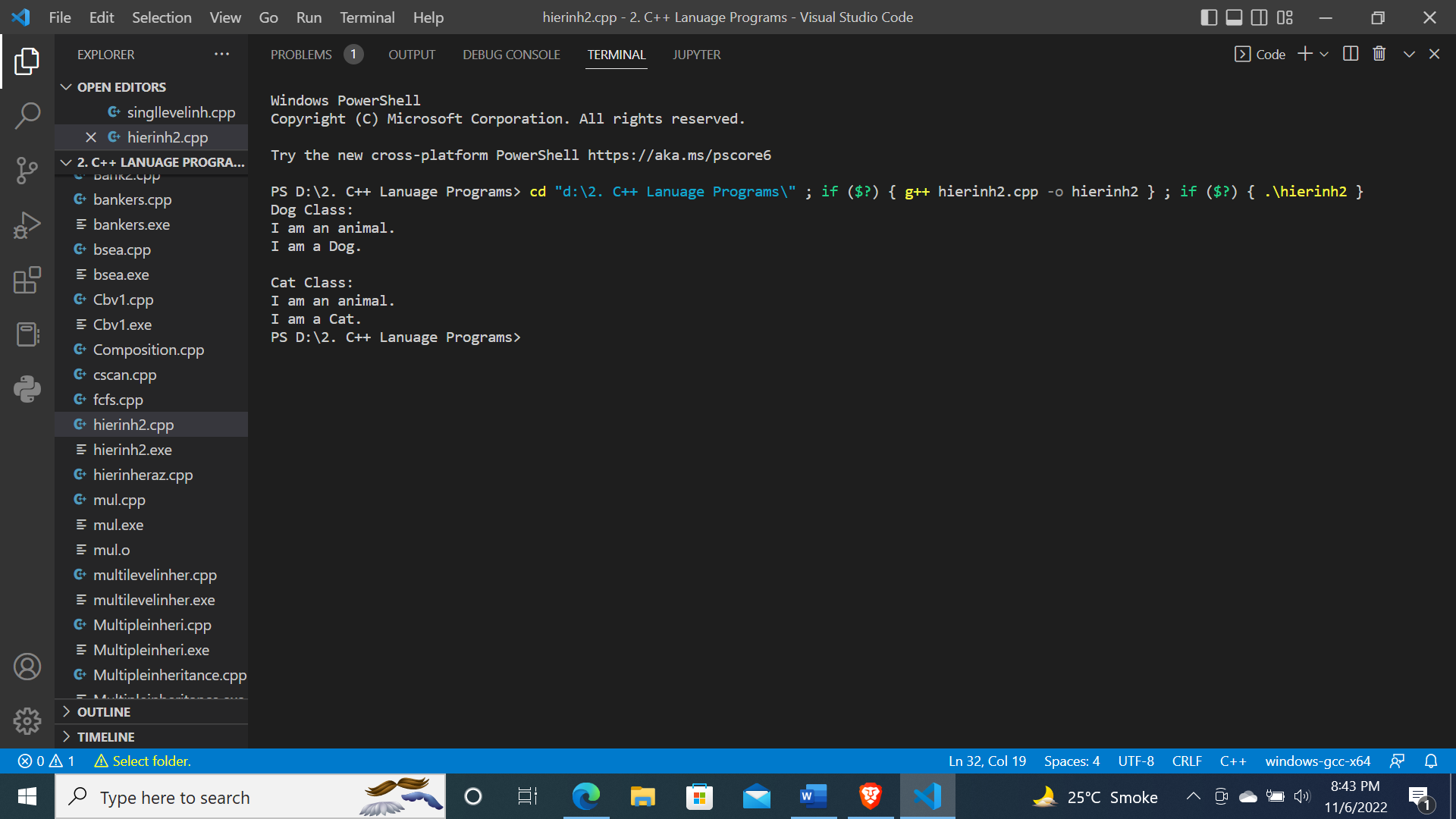
    cout << "\nCat Class:" << endl;

    cat1.info();

    cat1.meow();

    return 0;

}



**Program 9: Write a program which is showing hybrid inheritance.**

**Code:**

#include <iostream>

using namespace std;

class vehicle

{

    public:

        vehicle()

        {

            cout<< "This is a vehicle\n";

        }

};

class Car: public vehicle

{

    public:

        Car()

        {

            cout<< "This is a car\n";

        }

};

class Racing

{

    public:

        Racing()

        {

            cout<< "This is for Racing\n";

        }

};

class Ferrari: public Car, public Racing

{

    public:

        Ferrari()

        {

            cout<< "Ferrari is a Racing Car\n";

        }

};

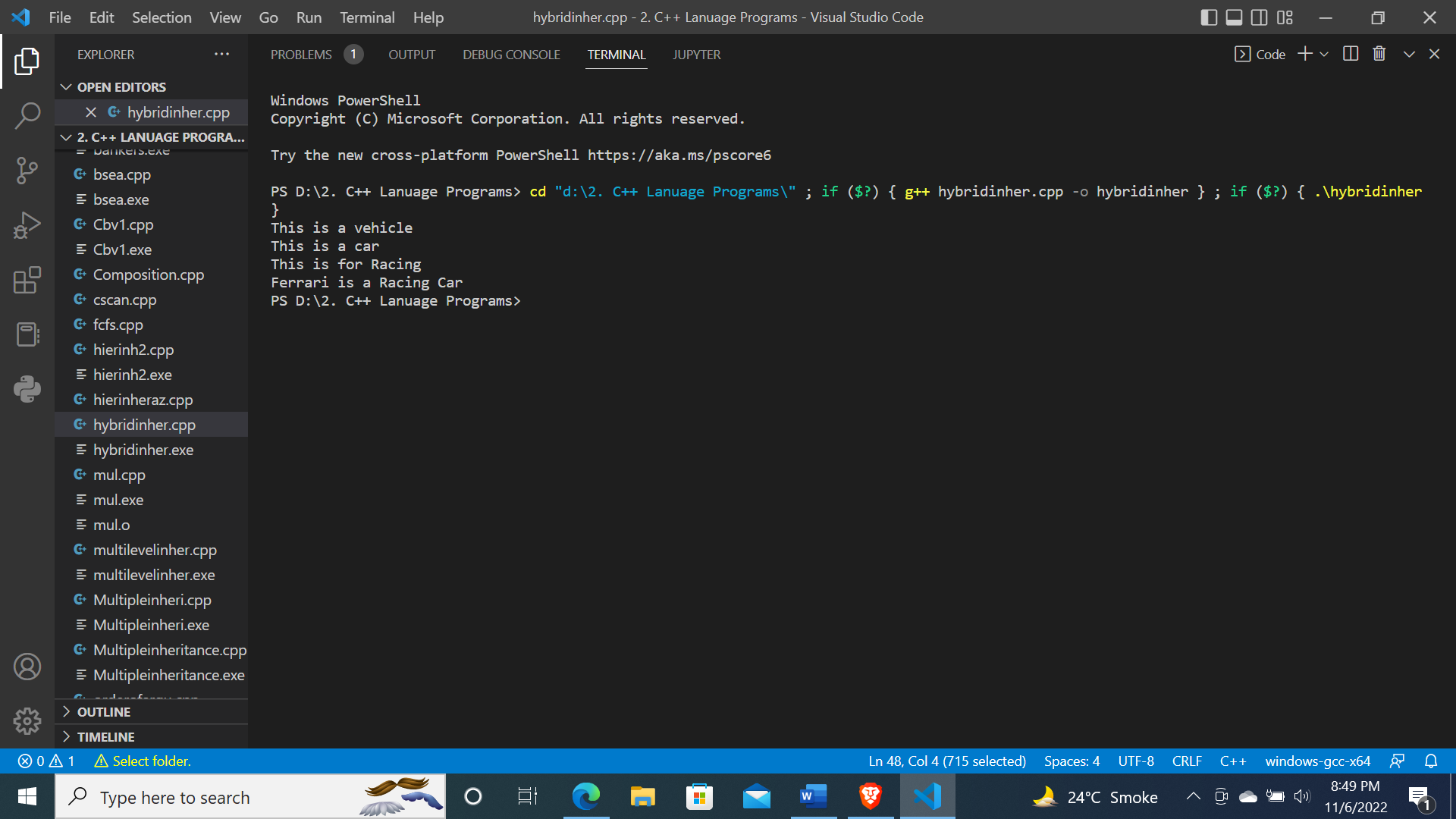
int main()

{

    Ferrari f;

    return 0;

}



**Program 10: Write a program showing inheritance with constructors and destructors.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

class A

{

public:

    A()

    {

        cout << "Constructor of A" << endl;

    }

    ~A()

    {

        cout << "Destructor of A" << endl;

    }

};

class B

{

public:

    B()

    {

        cout << "Constructor of B" << endl;

    }

    ~B()

    {

        cout << "Destructor of B" << endl;

    }

};

class C : public A

{

public:

    C()

    {

        cout << "Constructor of C" << endl;

    }

    ~C()

    {

        cout << "Destructor of C" << endl;

    }

};

class D : public A

{

public:

    D()

    {

        cout << "Constructor of D" << endl;

    }

    ~D()

    {

        cout << "Destructor of D" << endl;

    }

};

class E

{

public:

    E()

    {

        cout << "Constructor of E" << endl;

    }

    ~E()

    {

        cout << "Destructor of E" << endl;

    }

};

class F : public B, public C, public D

{

public:

    F()

    {

        cout << "Constructor of F" << endl;

    }

    ~F()

    {

        cout << "Destructor of F" << endl;

    }

};

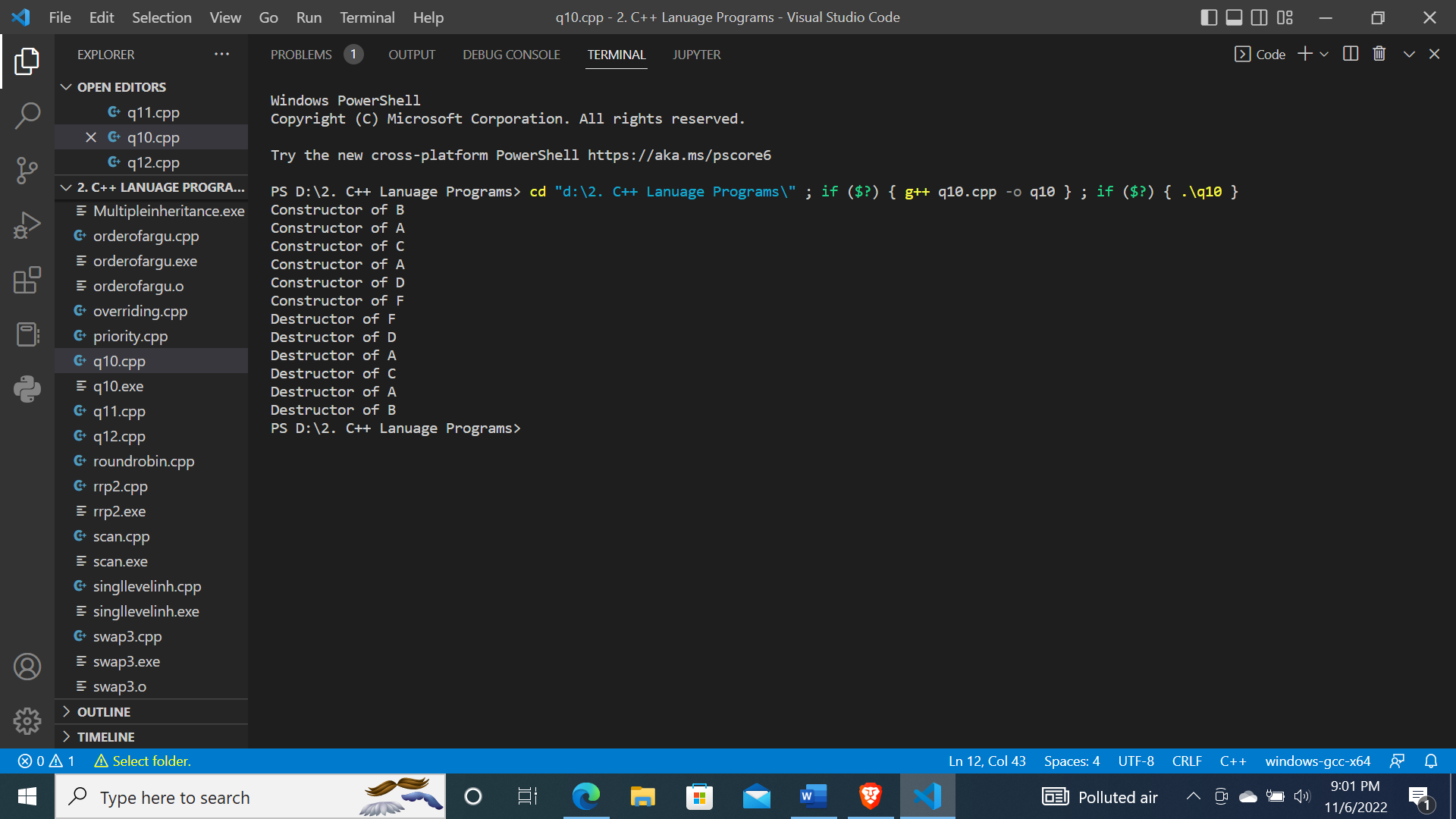
int main()

{

    F obj;

    return 0;

}



**Program 11: Write a program having copy constructor in it.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

class Emp

{

    int id = 1, a = 2;

public:

    Emp(int i, int j)

    {

        id = i;

        a = j;

    }

    Emp(Emp &e)

    {

        id = e.id;

    }

    void display()

    {

        cout << "Id = " << id << "\nA = " << a << endl;

    }

};

int main()

{

    cout << "Parametrized constructor:- " << endl;

    Emp e1(10, 11);

    e1.display();

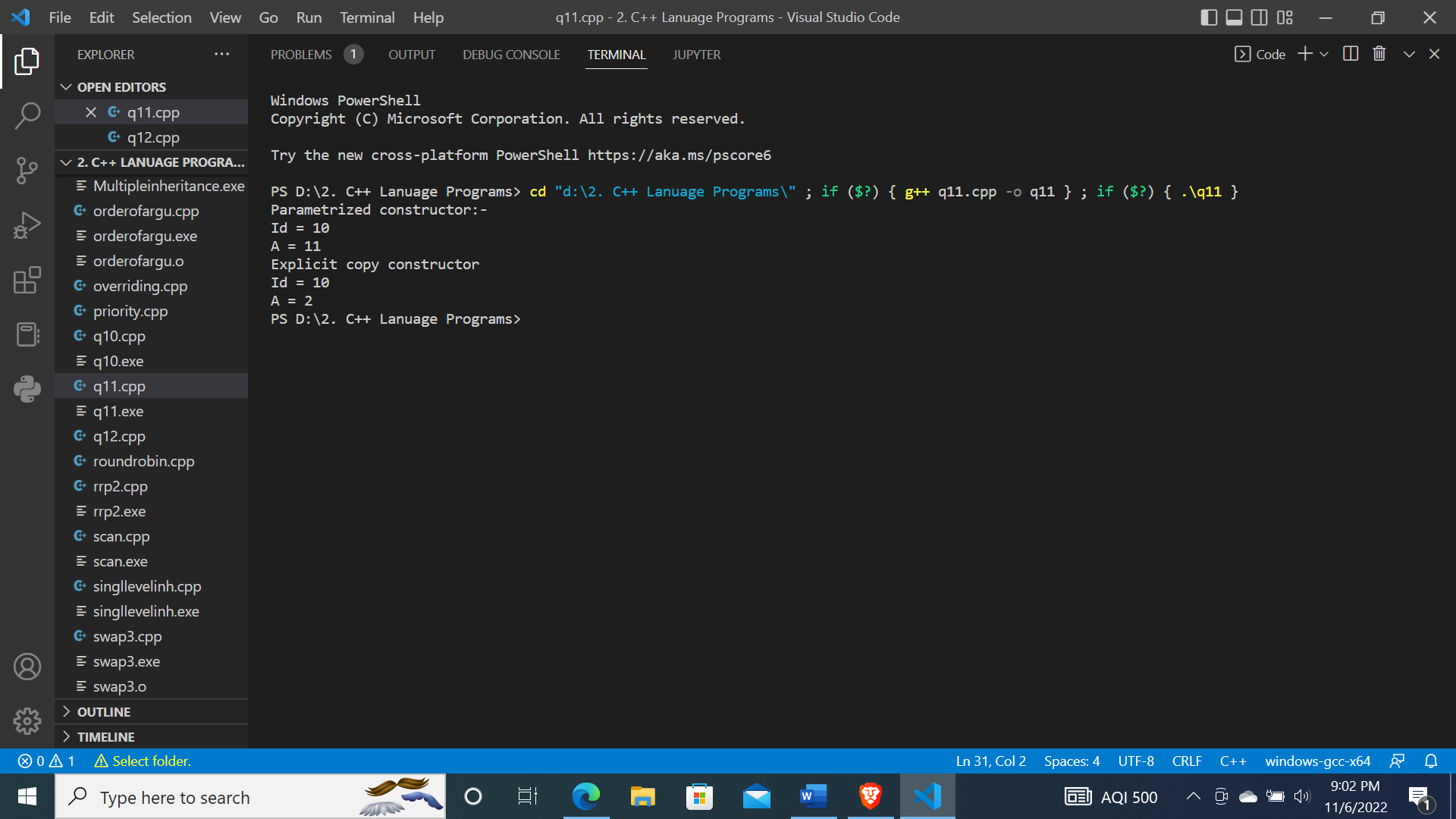
    cout << "Explicit copy constructor" << endl;

    Emp e2(e1);

    e2.display();

    return 0;

}



**Program 12: Write a program of inheritance of classes using virtual keyword.**

**Code:**

#include<bits/stdc++.h>

using namespace std;

class A{

    public:

    void display(){

        cout<<"A class (Base Class)"<<endl;

    }

};

class B:virtual public A {

    public:

    void display(){

        cout<<"B class (Derived from class A)"<<endl;

    }

};

class C:virtual public A {

    public:

    void display(){

        cout<<"C class (Derived from class A)"<<endl;

    }

};

class D:public B, public C{

    public:

    void display(){

        cout<<"D class (Derived from class B and C)"<<endl;

    }

};

int main(){

    D d;

    d.display();

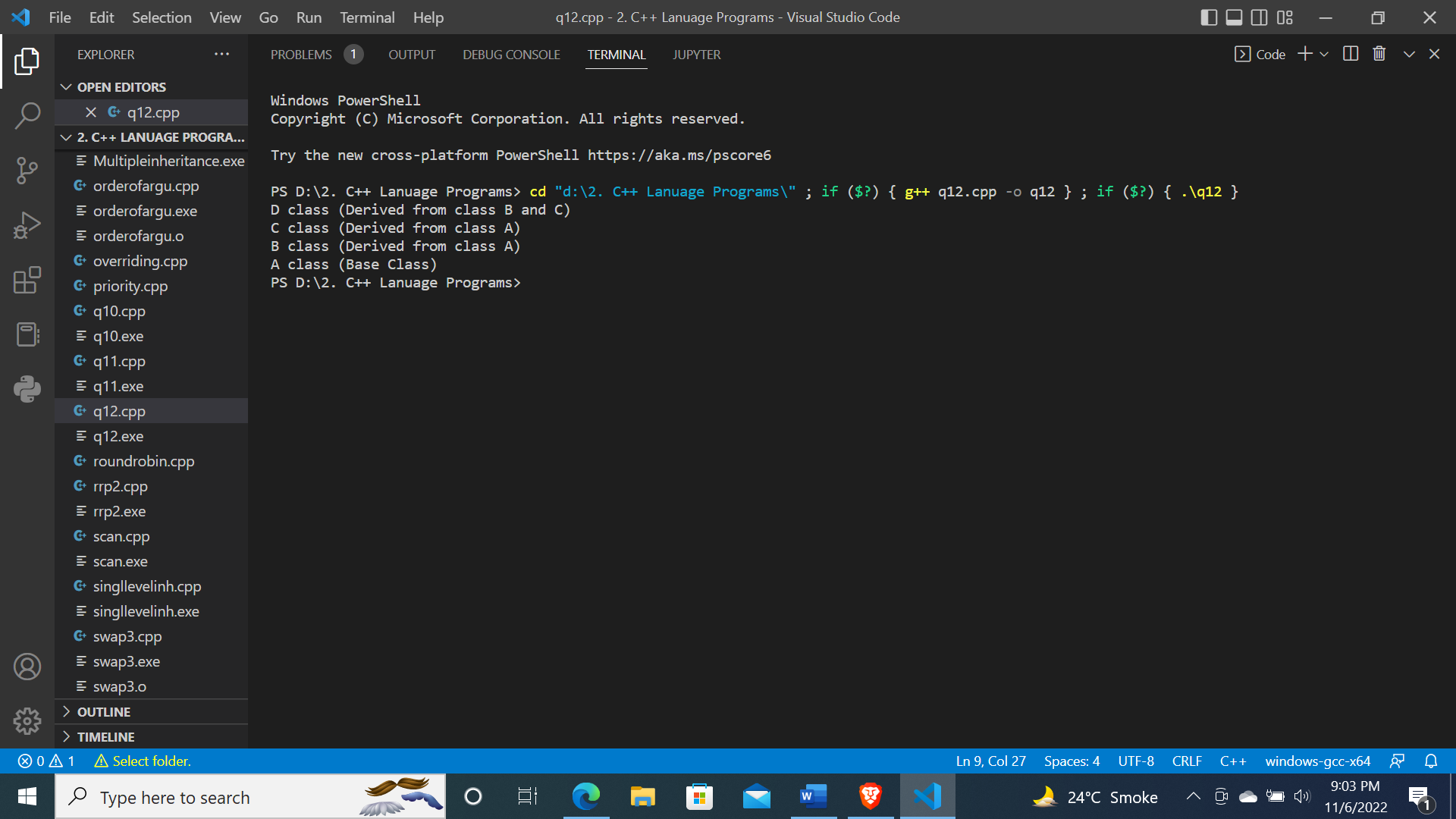
    d.C::display();

    d.B::display();

    d.A::display();

    return 0;

}



**Program 13: Write a program to implement function overloading which having arguments of different datatype.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

class Rectangle

{

public:

    int perimeter(int, int);

    double perimeter(double, double);

};

int Rectangle ::perimeter(int a, int b)

{

    return 2 \* (a + b);

}

double Rectangle ::perimeter(double a, double b)

{

    return 2 \* (a + b);

}

int main()

{

    Rectangle r1;

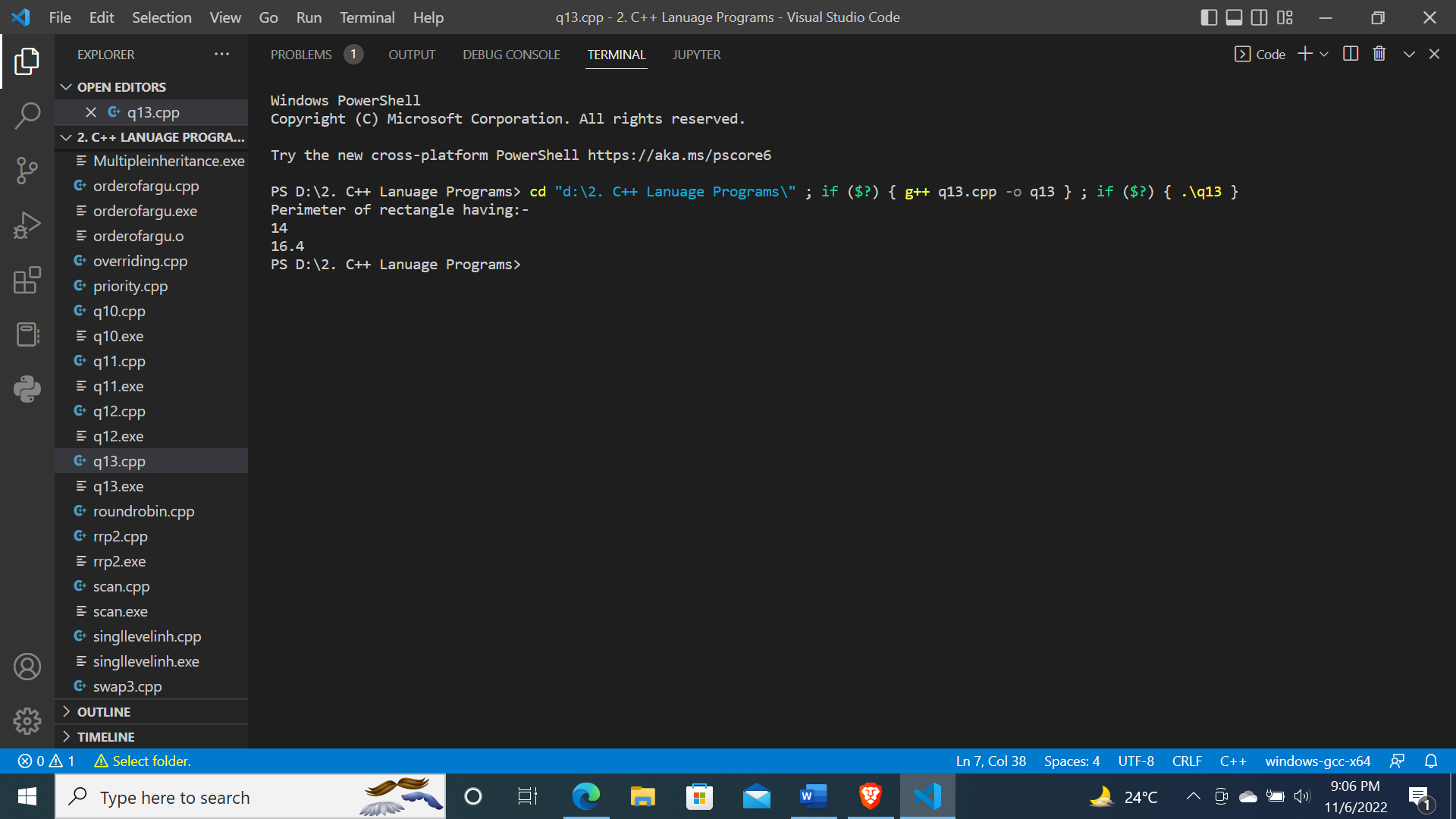
    cout << "Perimeter of rectangle having:- "<<endl;

    cout << r1.perimeter(2, 5) << endl;

    cout << r1.perimeter(2.5, 5.7);

    return 0;

}



**Program 14: Write a program to implement function overloading which having different number of arguments.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

class Overload

{

public:

    void area(int r)

    {

        cout << "Area of circle:- " << 3.14 \* pow(r, 2) << endl;

    }

    void area(int l, int b)

    {

        cout << "Area of rectangle:- " << l \* b << endl;

    }

} o1;

int main()

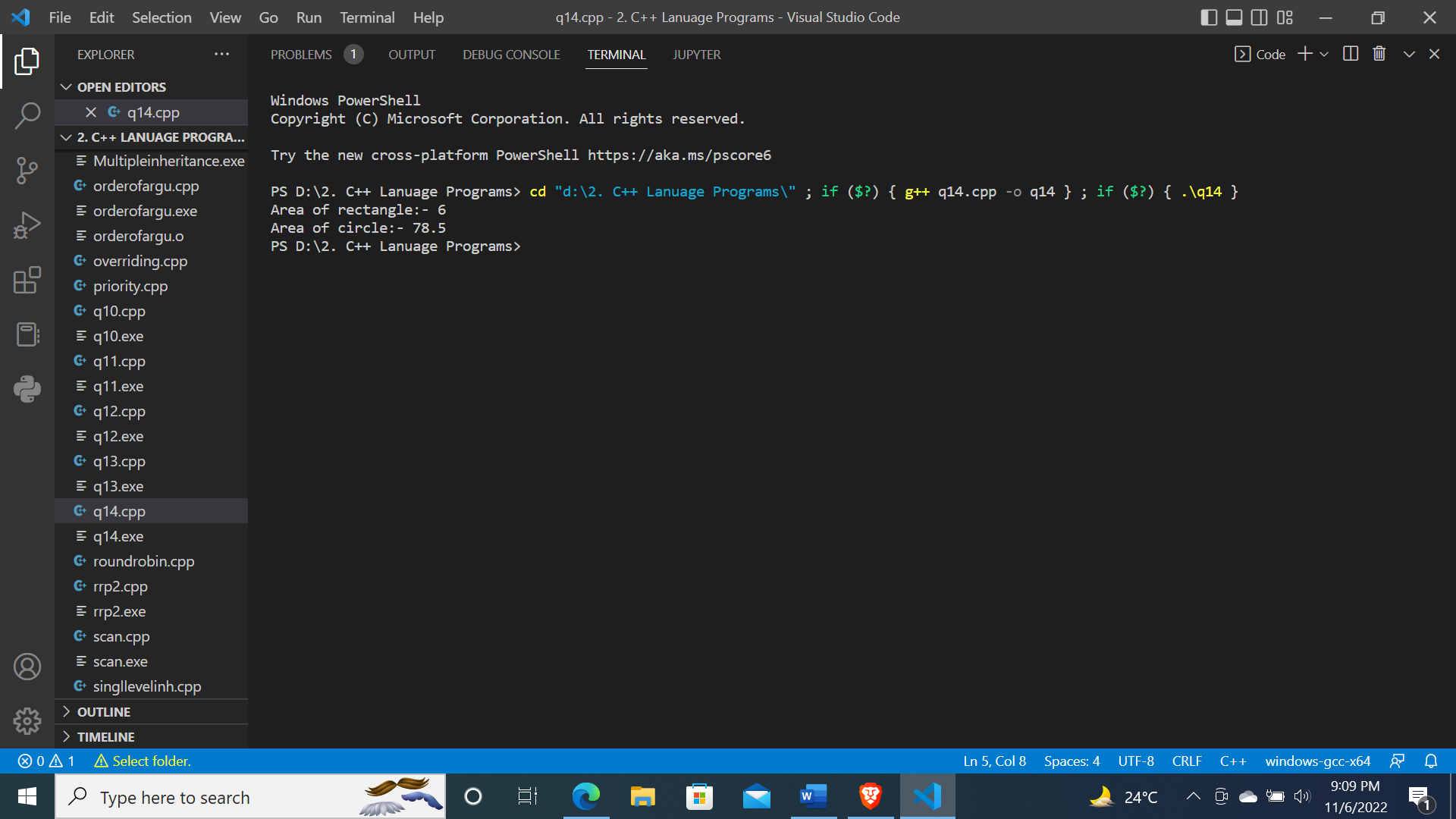
{

    o1.area(2, 3);

    o1.area(5);

    return 0;

}



**Program 15: Write a program to implement function overloading which having different order of arguments.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

class Overload

{

public:

    void func(int a, float b)

    {

        cout << "Multiply:- " << a \* b;

    }

    void func(float b, int a)

    {

        cout << "Add:- " << a + b;

    }

} o1;

int main()

{

    o1.func(2, 2.5);

    cout << endl;

    o1.func(2.5, 2);

    cout << endl;

    return 0;

}

