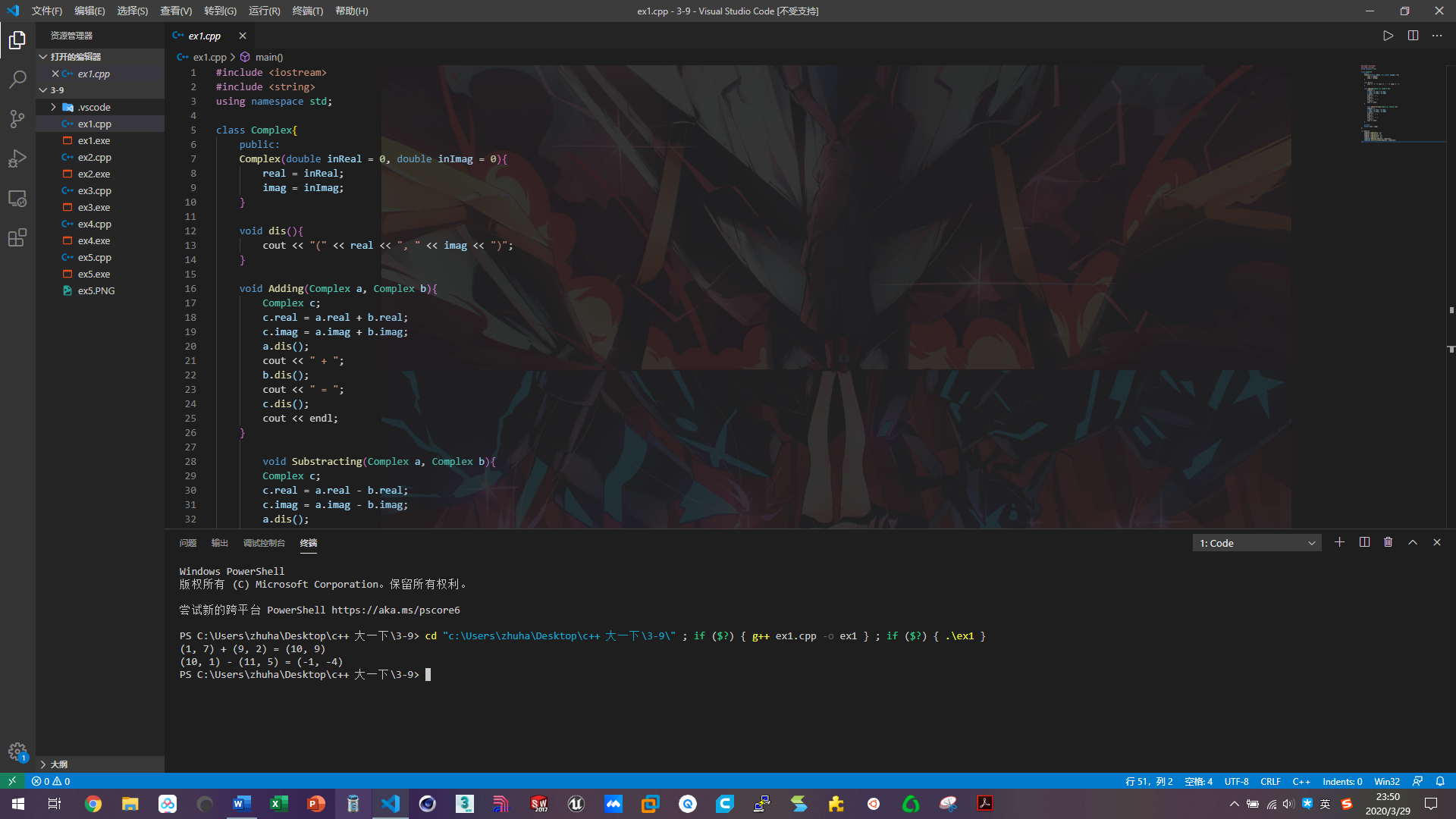
Ex1



#include <iostream>

#include <string>

using namespace std;

class Complex{

    public:

    Complex(double inReal = 0, double inImag = 0){

        real = inReal;

        imag = inImag;

    }

    void dis(){

        cout << "(" << real << ", " << imag << ")";

    }

    void Adding(Complex a, Complex b){

        Complex c;

        c.real = a.real + b.real;

        c.imag = a.imag + b.imag;

        a.dis();

        cout << " + ";

        b.dis();

        cout << " = ";

        c.dis();

        cout << endl;

    }

        void Substracting(Complex a, Complex b){

        Complex c;

        c.real = a.real - b.real;

        c.imag = a.imag - b.imag;

        a.dis();

        cout << " - ";

        b.dis();

        cout << " = ";

        c.dis();

        cout << endl;

    }

    private:

    double real, imag;

};

int main(){

    Complex complex1(1, 7);

    Complex complex2(9, 2);

    Complex complex3(10, 1);

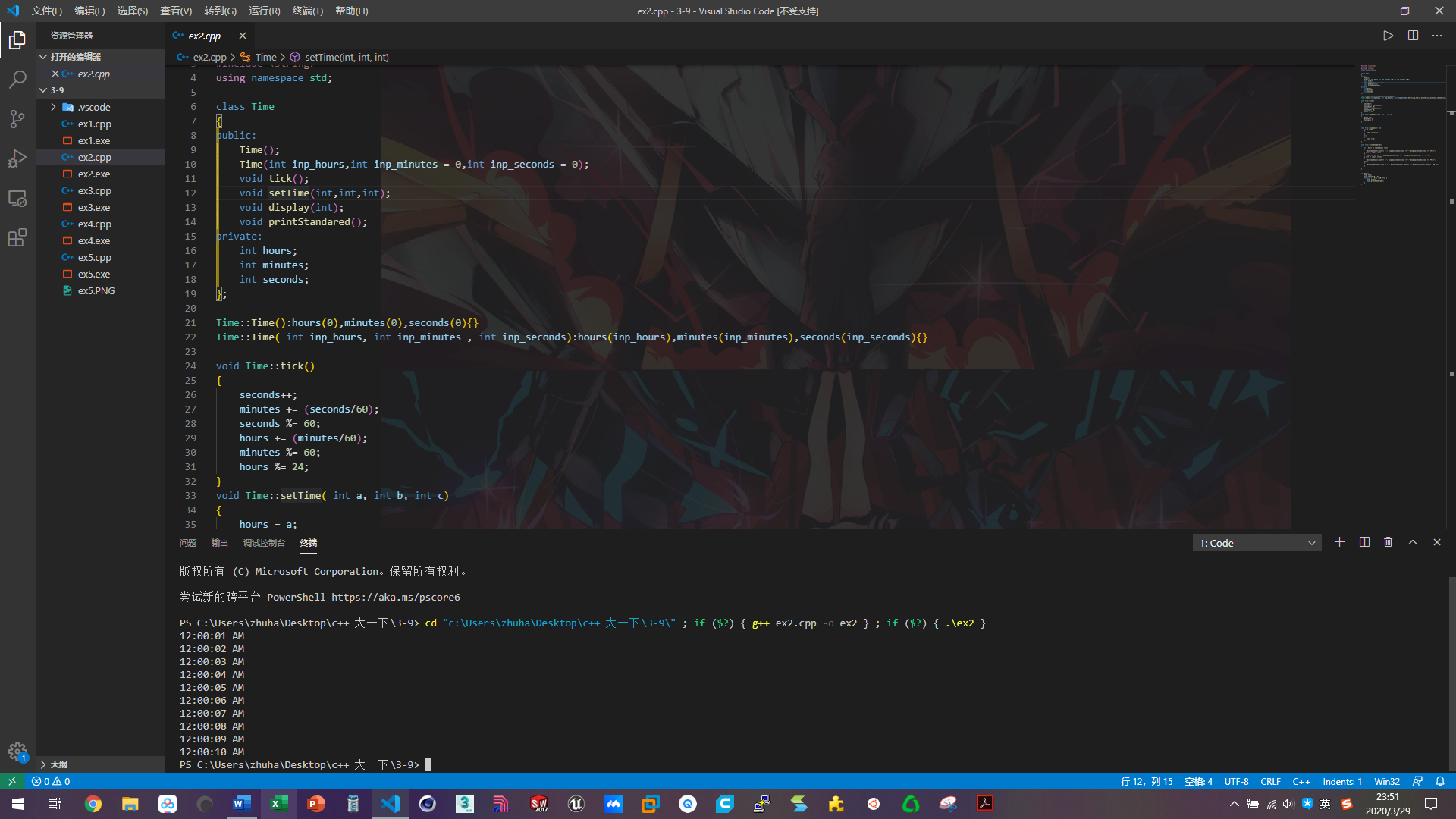
    Complex complex4(11, 5);

    complex1.Adding(complex1, complex2);

    complex3.Substracting(complex3, complex4);

}

Ex2



#include <iostream>

#include <ctime>

#include <string>

using namespace std;

class Time

{

public:

    Time();

    Time(int inp\_hours,int inp\_minutes = 0,int inp\_seconds = 0);

    void tick();

    void setTime(int,int,int);

    void display(int);

    void printStandared();

private:

    int hours;

    int minutes;

    int seconds;

};

Time::Time():hours(0),minutes(0),seconds(0){}

Time::Time( int inp\_hours, int inp\_minutes , int inp\_seconds):hours(inp\_hours),minutes(inp\_minutes),seconds(inp\_seconds){}

void Time::tick()

{

    seconds++;

    minutes += (seconds/60);

    seconds %= 60;

    hours += (minutes/60);

    minutes %= 60;

    hours %= 24;

}

void Time::setTime( int a, int b, int c)

{

    hours = a;

    minutes = b;

    seconds = c;

}

void Time::display(int a){

    if (a < 10)

    {

        cout << "0" << a;

    }

    else

    {

        cout << a;

    };

}

void Time::printStandared()

{

    if ( hours >= 1 and hours < 12)

    {

        display(hours);cout << ":"; display(minutes);cout << ":";display(seconds);cout <<" AM \n";

    } else if (hours == 0)

    {

        cout << "12" << ":"; display(minutes);cout << ":";display(seconds);cout <<" AM \n";

    } else if (hours == 12)

    {

        display(hours);cout << ":"; display(minutes);cout << ":";display(seconds);cout <<" PM \n";

    }else

    {

        display(hours%12);cout << ":"; display(minutes);cout << ":";display(seconds);cout << " PM \n";

    }

}

int main() {

    Time time;

    time.setTime(0,0,0);

    for (int i = 0; i < 10; ++i) {

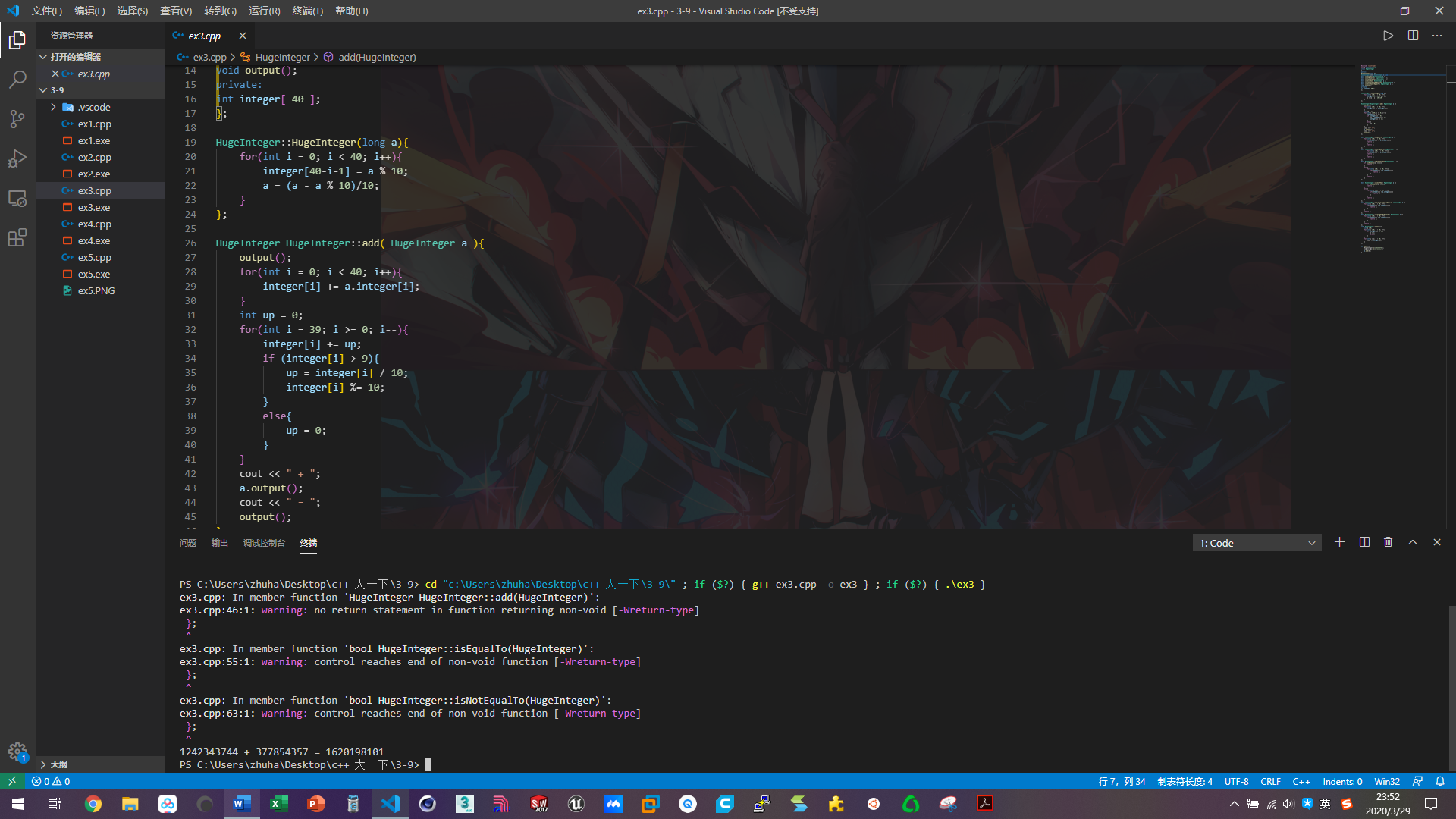
        time.tick();

        time.printStandared();

    }

}

Ex3



#include <iostream>

using namespace std;

class HugeInteger

{

public:

HugeInteger(long a);

HugeInteger add( HugeInteger a );

bool isEqualTo( HugeInteger a );

bool isNotEqualTo( HugeInteger a );

bool isGreaterThan(HugeInteger a );

bool isLessThan( HugeInteger a );

bool isGreaterThanOrEqualTo( HugeInteger a );

bool isLessThanOrEqualTo( HugeInteger a );

void output();

private:

int integer[ 40 ];

};

HugeInteger::HugeInteger(long a){

    for(int i = 0; i < 40; i++){

        integer[40-i-1] = a % 10;

        a = (a - a % 10)/10;

    }

};

HugeInteger HugeInteger::add( HugeInteger a ){

    output();

    for(int i = 0; i < 40; i++){

        integer[i] += a.integer[i];

    }

    int up = 0;

    for(int i = 39; i >= 0; i--){

        integer[i] += up;

        if (integer[i] > 9){

            up = integer[i] / 10;

            integer[i] %= 10;

        }

        else{

            up = 0;

        }

    }

    cout << " + ";

    a.output();

    cout << " = ";

    output();

};

bool HugeInteger::isEqualTo( HugeInteger a ){

    for(int i = 0; i < 40; i++){

        if(integer[i] != a.integer[i]){

        return 0;

        }

        return 1;

    }

};

bool HugeInteger::isNotEqualTo( HugeInteger a ){

        for(int i = 0; i < 40; i++){

        if(integer[i] != a.integer[i]){

        return 1;

        }

        return 0;

    }

};

bool HugeInteger::isGreaterThan(HugeInteger a ){

    if (isEqualTo(a) == 1){

        return 0;

    }

    else{

        for(int i = 0; i < 40; i++){

            if(integer[i] < a.integer[i]){

                return 0;

            }

        }

        return 1;

    }

};

bool HugeInteger::isLessThan( HugeInteger a ){

        if (isEqualTo(a) == 1){

        return 0;

    }

    else{

        for(int i = 0; i < 40; i++){

            if(integer[i] > a.integer[i]){

                return 0;

            }

        }

        return 1;

    }

};

bool HugeInteger::isGreaterThanOrEqualTo( HugeInteger a ){

    for(int i = 0; i < 40; i++){

        if(integer[i] < a.integer[i]){

            return 0;

        }

    }

    return 1;

};

bool HugeInteger::isLessThanOrEqualTo( HugeInteger a ){

    for(int i = 0; i < 40; i++){

        if(integer[i] > a.integer[i]){

            return 0;

        }

    }

    return 1;

};

void HugeInteger::output(){

    int a = 0;

    for(int i = 0; i < 40; i++){

        if (integer[i] != 0){

            a = i;

            break;

        }

    }

    for(int j = a; j < 40; j++){

        cout << integer[j];

    }

};

int main(){

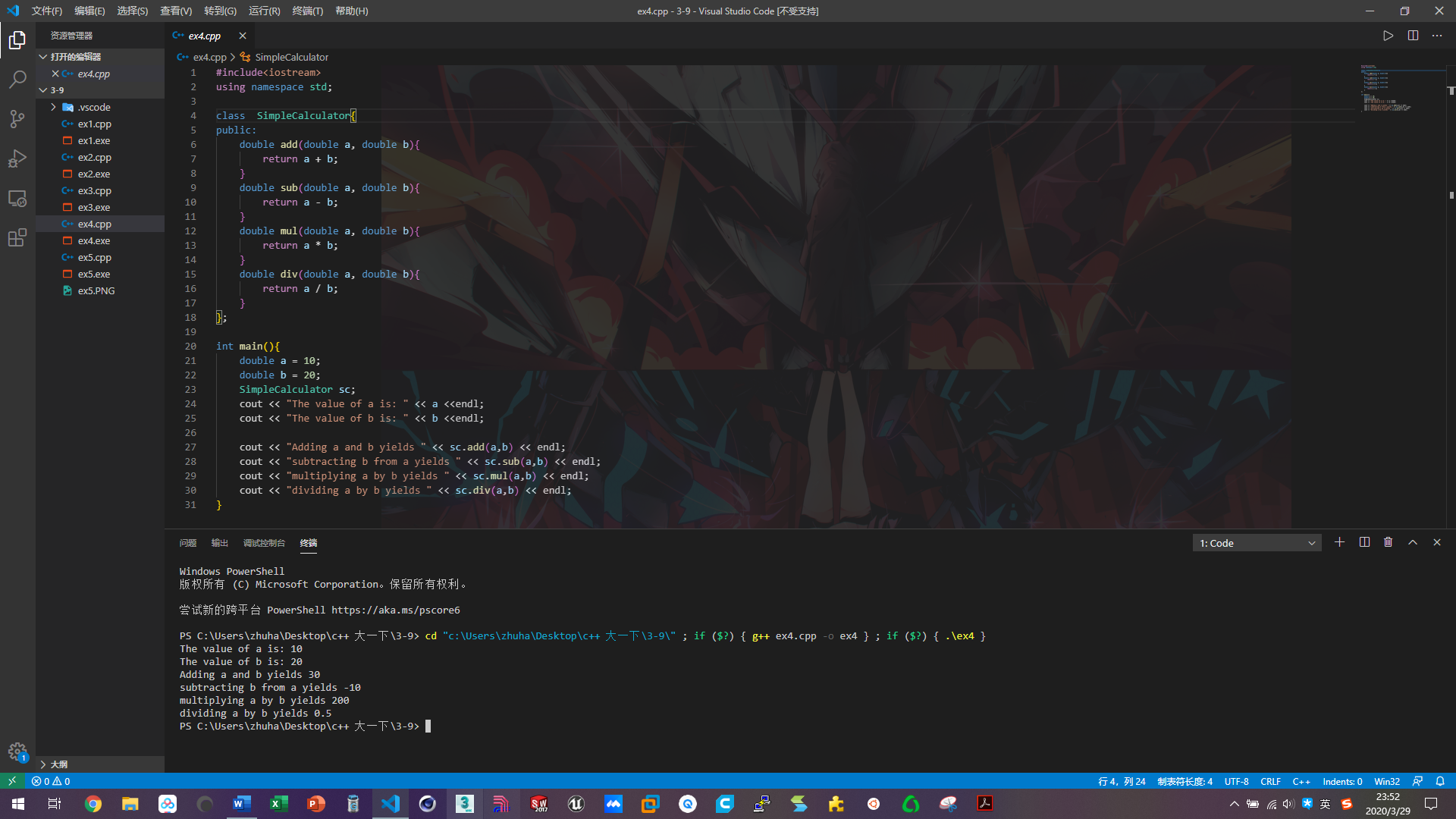
    HugeInteger a(1242343744);

    HugeInteger b(377854357);

    a.add(b);

}

Ex4



#include<iostream>

using namespace std;

class  SimpleCalculator{

public:

    double add(double a, double b){

        return a + b;

    }

    double sub(double a, double b){

        return a - b;

    }

    double mul(double a, double b){

        return a \* b;

    }

    double div(double a, double b){

        return a / b;

    }

};

int main(){

    double a = 10;

    double b = 20;

    SimpleCalculator sc;

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

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

    cout << "Adding a and b yields " << sc.add(a,b) << endl;

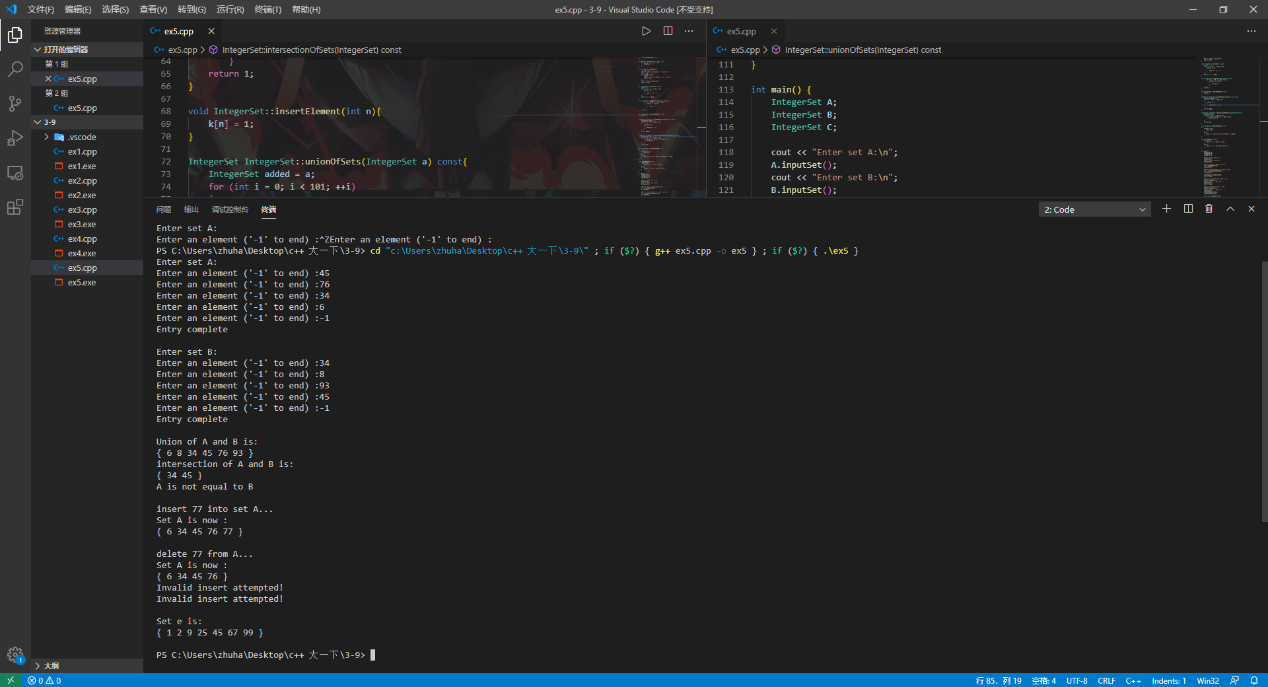
    cout << "subtracting b from a yields " << sc.sub(a,b) << endl;

    cout << "multiplying a by b yields " << sc.mul(a,b) << endl;

    cout << "dividing a by b yields " << sc.div(a,b) << endl;

}

Ex5



#include <iostream>

using namespace std;

class IntegerSet{

private:

    int k[101];

public:

    IntegerSet()

    {

        for(int i = 0;i < 101;++i)

        {

            k[i] = 0;

        }

    }

    IntegerSet(int\*,int);

    void inputSet();

    void printSet() const;

    bool isEqualTo(IntegerSet) const;

    void insertElement(int);

    IntegerSet unionOfSets(IntegerSet) const;

    IntegerSet intersectionOfSets(IntegerSet) const;

    void deleteElement(int);

};

IntegerSet::IntegerSet(int\* num,int n){

    for (int i = 0; i < n; ++i) {

        k[num[i]] = 1;

    }

}

void IntegerSet::inputSet(){

    int inp = 0;

    cout << "Enter an element ('-1' to end) :";

    cin >> inp;

    while (inp != -1){

        k[inp] = 1;

        cout << "Enter an element ('-1' to end) :";

        cin >> inp;

    }

    cout << "Entry complete\n";

    cout << endl;

}

void IntegerSet::printSet() const{

    cout << "{ ";

    for (int i = 0; i < 101; ++i){

        if(k[i] == 1)

        {

            cout << i << " ";

        }

    }

    cout << "}" << endl;

}

bool IntegerSet::isEqualTo(IntegerSet a)const{

        for (int i = 0; i < 101; ++i){

            if(k[i] != a.k[i])

            {

                return 0;

            }

        }

    return 1;

}

void IntegerSet::insertElement(int n){

    k[n] = 1;

}

IntegerSet IntegerSet::unionOfSets(IntegerSet a) const{

    IntegerSet added = a;

    for (int i = 0; i < 101; ++i)

    {

        if(k[i] == 1)

        {

            added.k[i] = 1;

        }

    }

    return added;

}

IntegerSet IntegerSet::intersectionOfSets(IntegerSet a)const{

    IntegerSet iso;

    for (int i = 0; i < 101; ++i){

        if(a.k[i] == 1 && this -> k[i] == 1){

            iso.k[i] = 1;

        }

    }

    return iso;

}

void IntegerSet::deleteElement(int n)

{

    if(k[n] == 1){

        k[n] = 0;

    }

    else{

        cout << "Invalid insert attempted!" << endl;

    }

}

void printEqual(bool a){

    if(a == 1){

        cout << "A is equal to B\n";

    }

    else{

        cout << "A is not equal to B\n";

    }

}

int main() {

    IntegerSet A;

    IntegerSet B;

    IntegerSet C;

    cout << "Enter set A:\n";

    A.inputSet();

    cout << "Enter set B:\n";

    B.inputSet();

    cout << "Union of A and B is:\n";

    C = A.unionOfSets(B);

    C.printSet();

    cout << "intersection of A and B is:\n";

    C = A.intersectionOfSets(B);

    C.printSet();

    printEqual(A.isEqualTo(B));

    cout << endl;

    cout << "insert 77 into set A...\n";

    A.insertElement(77);

    cout << "Set A is now :\n";

    A.printSet();

    cout << endl;

    cout << "delete 77 from A...\n";

    A.deleteElement(77);

    cout << "Set A is now :\n";

    A.printSet();

    A.deleteElement(101);

    A.deleteElement(50);

    const int arraySize = 10;

    int intArray[arraySize] = {25,67,2,9,99,105,45,-5,101,1};

    IntegerSet e(intArray,arraySize);

    cout << "\nSet e is:\n";

    e.printSet();

    cout << endl;

    return 0;

}