第一题

# coding=gbk  
from fractions import Fraction  
import numpy as np  
  
try:  
 np.set\_printoptions(formatter={'all': lambda x: str(Fraction(x).limit\_denominator())})  
  
 m = int(input("输入矩阵行数:\n"))  
 A = [[] for i in range(m)]  
 print("请输入矩阵元素，同一行使用空格进行分隔，输入回车切换到下一行:")  
  
 for i in range(m):  
 a = input().split()  
 for j in range(len(a)):  
 A[i].append(a[j])  
  
 array\_before = np.array(A)  
 array\_before = np.array(array\_before, dtype='float')  
# 这一行是必须的，否则矩阵在计算的过程中类型会变成object，无法求逆  
 array\_after = np.linalg.inv(array\_before)  
except Exception:  
 print("矩阵不可逆")  
else:  
 print("逆矩阵为:\n", array\_after)

第二题

#include<iostream>

#include<iomanip>

using namespace std;

int const n = 3;

int main()

{

int aba;

cout << "输入矩阵的阶数a" << endl;

cin >> aba;

void temp(double aa[], double bb[], int n);

double fun(double array[n][n]);

double a[n][n], b[n][2 \* n], c[n][n], det1, yinzhi;

double bb;

int i, j, kk = 0, k, u;

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

for (j = 0; j < 2 \* n; j++)

b[i][j] = 0;

cout << "请输入一个" << n << "节方阵" << endl;

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

for (j = 0; j < n; j++)

cin >> a[i][j];

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

for (j = 0; j < n; j++)

b[i][j] = a[i][j];

for (j = 0; j < n; j++)

b[j][n + j] = 1;

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

{

if (b[i][i] == 0)

for (j = i; j < n; j++)

{

if (b[j][i] != 0)

temp(b[i], b[j], 2 \* n);

}

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

{

yinzhi = -1 \* b[k][i] / b[i][i];

for (u = 0; u < 2 \* n; u++)

{

b[k][u] = b[k][u] + b[i][u] \* yinzhi;

}

}

}

det1 = fun(a);

if (det1 == 0)

{

cout << "此矩阵不可逆：" << endl;

return 0;

}

if (det1 != 0)

{

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

{

bb = b[i][i];

for (j = 0; j < 2 \* n; j++)

b[i][j] = b[i][j] / bb;

}

for (i = n - 1; i > 0; i--)

for (k = 0; k < i; k++)

{

bb = b[k][i];

for (u = 0; u < 2 \* n; u++)

b[k][u] = b[k][u] - bb \* b[i][u];

}

}

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

for (j = 0; j < n; j++)

c[i][j] = b[i][j + n];

kk = 0;

if (det1 != 0)

{

cout << "其可逆且其行列式的值det为：" << det1 << endl << endl;

cout << "可逆a矩阵的逆矩阵为c矩阵：" << endl;

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

for (j = 0; j < n; j++)

{

cout << setw(15) << c[i][j];

kk = kk + 1;

if (kk % n == 0)

cout << endl;

}

}

return 0;

}

void temp(double aa[], double bb[], int n)

{

int i;

double temp1;

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

{

temp1 = aa[i];

aa[i] = bb[i];

bb[i] = temp1;

}

}

double fun(double array[n][n])

{

int ii, jj, k, u;

int iter = 0;

double det1 = 1, yin;

for (ii = 0; ii < n; ii++)

{

if (array[ii][ii] == 0)

for (jj = ii; jj < n; jj++)

{

if (array[jj][ii] != 0)

{

temp(array[ii], array[jj], n);//交换两行

iter++;

}

}

for (k = ii + 1; k < n; k++)

{

yin = -1 \* array[k][ii] / array[ii][ii];

for (u = 0; u < n; u++)

{

array[k][u] = array[k][u] + array[ii][u] \* yin;

}

}

}

for (ii = 0; ii < n; ii++)

det1 = det1 \* array[ii][ii];

if (iter % 2 == 1)

det1 = -det1;

return (det1); //返回行列式的值

}

第四题

#include<iostream>

using namespace std;

int main() {

int a;

cout << "请选择你要输入的数字类型" << endl;

cout << "1--整数" << endl;

cout << "2--小数" << endl;

cout << "3--分数" << endl;

cout << "4--百分数" << endl;

cin >> a;

switch (a) {

case 1:

int n;

cout << "请输入一个整数" << endl;

cin >> n;

while (n) {

printf("%d", n % 10);

n /= 10;

}

break;

}

return 0;

}

第五题

#include <iostream>

#include <cmath>

double a, b, c, d;

class Vector

{

private:

double x;

double y;

public:

Vector(double x\_init, double y\_init)

{

x = x\_init, y = y\_init;

}

Vector add(Vector& v2)

{

return Vector(x + v2.x, y + v2.y);

}

void print()

{

std::cout << "(" << x << ", " << y << ")" << std::endl;

}

void dir()

{

double LEN = std::sqrt(x \* x + y \* y);

std::cout << "模长为： " << LEN << std::endl;

}

};

int main() {

std::cin >> a >> b >> c >> d;

Vector v1(a, b);

Vector v2(c, d);

std::cout << "Vector v1: ";

v1.print();

std::cout << "Vector v2: ";

v2.print();

Vector v3 = v1.add(v2);

std::cout << "Vector v3: ";

v3.print();

std::cout << "v3 ";

v3.dir();

return 0;

}

第六题

PY

import socket  
  
s = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)  
  
s.connect(('127.0.0.1',12345))  
  
while True:  
 str = input("#:3")  
 str2 = input("#:4")  
 if(str == ""):  
 break  
  
 s.send(str.encode('utf-8'))  
 s.send(str2.encode('utf-8'))  
  
s.close()

C

#include <iostream>

#include <boost/asio.hpp>

#include <boost/bind/bind.hpp>

#include <cmath>

using boost::asio::ip::tcp;

// 定义一个服务器类

class Server {

public:

// 构造函数，初始化服务器并开始接受连接

Server(boost::asio::io\_context& io\_context, short port)

: acceptor\_(boost::asio::make\_strand(io\_context), tcp::endpoint(tcp::v4(), port)) {

start\_accept();

}

private:

// 开始接受连接

void start\_accept() {

// 创建一个新的 socket

auto socket = std::make\_shared<tcp::socket>(acceptor\_.get\_executor());

// 异步接受连接

acceptor\_.async\_accept(\*socket, boost::bind(&Server::handle\_accept, this, socket, boost::asio::placeholders::error));

}

// 处理接受到的连接

void handle\_accept(std::shared\_ptr<tcp::socket> socket, const boost::system::error\_code& error) {

if (!error) {

// 读取客户端发送的数据

char data[1024];

size\_t length = socket->read\_some(boost::asio::buffer(data));

std::string message(data, length);

// 解析数据，假设数据是两个浮点数

double a, b;

sscanf(message.c\_str(), "%lf %lf", &a, &b);

// 计算两个数的平方和的平方根

// 这一段自己补充！

double h1,h2,response;

h1 = a \* a;

h2 = b \* b;

response = sprt(h1 + h2);

// 将结果转换为字符串并发送回客户端

std::string response = std::to\_string(c);

boost::asio::write(\*socket, boost::asio::buffer(response));

}

// 继续接受下一个连接

start\_accept();

}

// 接受连接的对象

tcp::acceptor acceptor\_;

};

int main() {

try {

// 创建 io\_context 对象

boost::asio::io\_context io\_context;

// 创建服务器对象，监听端口 12345（重要）

Server server(io\_context, 12345);

// 运行 io\_context，开始处理异步操作

io\_context.run();

}

catch (std::exception& e) {

// 捕获并打印异常

std::cerr << "Exception: " << e.what() << "\n";

}

return 0;

}