Task 1:

#include<iostream>

using namespace std;

class calarea

{

private:

int length;

int width;

int area;

static int count;

public:

void arecount(int l, int w);

int ansarea(int length, int width);

void display()

{

cout << "\n area is : " << area<<"\n count is : "<<count;

}

};

int calarea::count = 0;

void calarea::arecount(int l, int w)

{

length = l;

width = w;

}

int calarea::ansarea(int length, int width)

{

this->length = length;

this->width = width;

area = length \* width;

count++;

return area;

}

int main()

{

int num = 0;

int length = 0, width = 0,area=0;

calarea a;

a.arecount(0, 0);

cout << "enter any num except 0 to enter values of area : ";

cin >> num;

while (num!=0)

{

cout << "enter length : ";

cin >> length;

cout << endl << "enter width : ";

cin >> width;

cout<< a.ansarea(length, width);

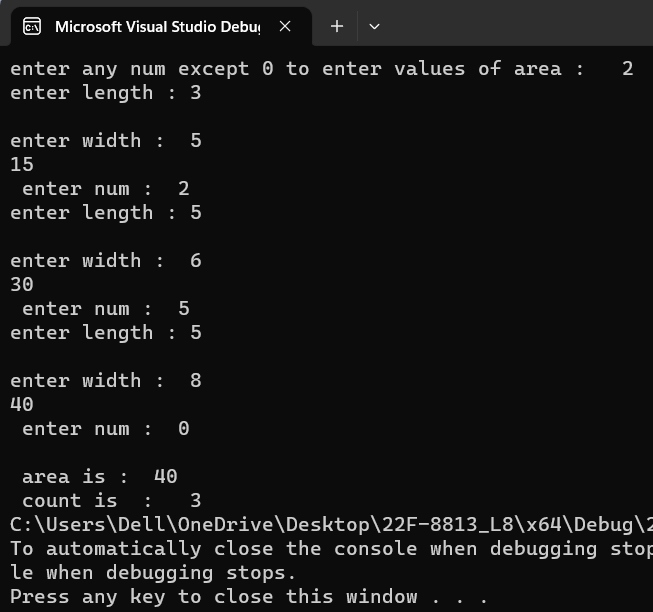
cout << "\n enter num : ";

cin >> num;

}

a.display();

}



Task 2:

#include<iostream>

using namespace std;

class matrix

{

private:

int row;

int col;

int a1[2][3];

int a2[2][3];

int a3[2][3];

public:

void constructor(int row, int col, int \*\*ptr,int\*\*ptr3);

void inputdata(int row,int col,int \*\*ptr1);

void addmat();

//~matrix();

};

void matrix::addmat()

{

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 3; j++)

{

a3[i][j] = a1[i][j] + a2[i][j];

}

}

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 3; j++)

{

cout << a3[i][j]<<" ";

}

cout << endl;

}

}

void matrix:: inputdata (int row,int col,int \*\*ptr1)

{

for (int i = 0; i < row; i++)

{

\*(ptr1 + i) = new int[3];

}

cout << "\nenter values \n ";

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

cin >> \*(\*(ptr1 + i) + j);

}

}

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

a2[i][j] = \*(\*(ptr1 + i) + j);

}

}

/\*for (int i = 0; i < 3; i++)

{

delete[] ptr1[i];

}delete[] ptr1;\*/

}

void matrix:: constructor(int row, int col, int \*\*ptr,int \*\*ptr3)

{

for (int i = 0; i < row; i++)

{

\*(ptr + i) = new int[3];

}

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

\*(\*(ptr + i) + j) = 1;

a1[i][j] = \*(\*(ptr + i) + j);

}

}

//for (int i = 0; i < 3; i++)

//{

// delete[] ptr[i];

//}

//delete[] ptr;

for (int i = 0; i < row; i++)

{

\*(ptr3 + i) = new int[3];

}

for (int i = 0; i < row; i++)

{

for (int j = 0; j < col; j++)

{

\*(\*(ptr3 + i) + j) = 0;

a3[i][j] = \*(\*(ptr3 + i) + j);

}

}

/\*for (int i = 0; i < 3; i++)

{

delete[] ptr3[i];

}delete[] ptr3;\*/

}

int main()

{

int\* ptr1 = new int[4];

\*(ptr1 + 0) = 0;

int\*\* pt = new int\* [2];

int\*\* ptr2 = new int\*[2];

int\*\* ptr3 = new int\* [2];

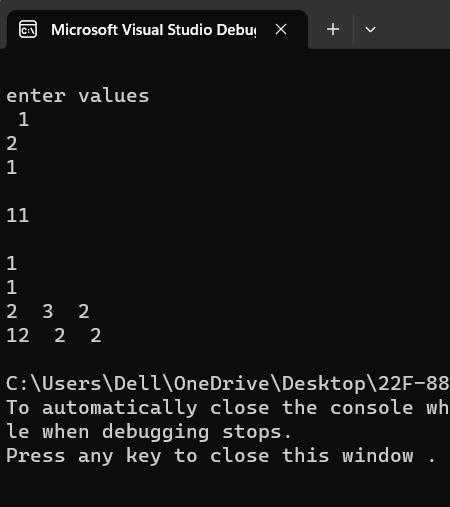
matrix obj;

obj.constructor(2, 3,ptr2,ptr3);

obj.inputdata(2,3,pt);

obj.addmat();

}



Task 4:

#include<iostream>

#include<string>

using namespace std;

class bank

{

private:

double cbal;

double intrate;

double intearn;

int trans;

static int count;

public:

void setdata(double cbal1, double intrate1);

void setintrate(double intrate2);

void makedeposit(double amountdep);

void withdraw(double amtwd);

void calcinterest()

{

intearn = cbal\*(intrate/100);

cbal+=intearn;

}

void countini(int count,int trans1);

void inccount()

{

count++;

}

double getcount() const

{

return count;

}

double const getinterestrate(double intrate);

double const getbalance(double cbal);

double const getinterest(double intearn);

int const gettransactions(int trans1);

};

int bank::count = 0;

void bank::countini(int c,int trans1)

{

trans = trans1;

}

void bank::setdata(double cbal1, double intrate1)

{

cbal = cbal1;

intrate = intrate1;

}

void bank::makedeposit(double amountdep)

{

cbal += amountdep;

trans++;

}

void bank:: withdraw(double amtwd)

{

if (amtwd < cbal)

{

cbal -=amtwd;

trans++;

}

else {

cout << "error";

}

}

double const bank:: getinterestrate(double intrate3)

{

cout << "\nenter interest rate : ";

cin >> intrate3;

intrate = intrate3;

return intrate;

}

double const bank::getbalance(double cbal1)

{

return cbal;

}

double const bank::getinterest(double intearn1)

{

return intearn;

}

int const bank::gettransactions(int trans1)

{

inccount();

trans += trans1;

return trans;

}

int main()

{

double temp = (4.5);

bank account;

account.countini(0,0);

account.setdata(0, temp);

int choice=0,amoun=0;

double amount=0, rate=0;

do {

cout << "1. Make a deposit" << endl;

cout << "2. Make a withdrawal" << endl;

cout << "3. Calculate interest" << endl;

cout << "4. Get current balance" << endl;

cout << "5. Get current interest rate" << endl;

cout << "6. Get current interest earned" << endl;

cout << "7. Get number of transactions" << endl;

cout << "8. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter the amount to deposit: ";

cin >> amount;

account.makedeposit( amount);

break;

case 2:

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

cin >> amount;

account.withdraw(amount);

break;

case 3:

account.calcinterest();

cout << "Interest calculated and added to the account." << endl;

break;

case 4:

cout << "Current balance: $" << account.getbalance(amount)<<endl;

break;

case 5:

cout << "Current interest rate: " << account.getinterestrate(amount) << "%" << endl;

break;

case 6:

cout << "Interest earned for the current period: $" << account.getinterest(amount) << endl;

break;

case 7:

cout << "Number of transactions for the current period: " << account.gettransactions(amoun) << endl;

break;

case 8:

cout << "Exiting program..." << endl;

break;

default:

cout << "Invalid choice. Please try again." << endl;

}

} while (choice != 8);

}

