

**National University of Computer and Emerging Sciences**

**Chiniot-Faisalabad Campus**

**BS (Artificial Intelligence)**

|  |  |
| --- | --- |
| **Name** | **M.Abdul Hanan** |
| **Reg.NO.** | **22F-3104** |
| **Section** | **BS(Ai)-A** |
| **Course** | **Object Oriented Analysis (CS 1004)** |
| **Department** | **CS Department** |
| **LAB** | **LAB Manual #14** |

**Task no 1:**

#include <iostream>

using namespace std;

int a = 1, b = 2, c = 3;

class COUNTING {

public:

int count=1;

COUNTING() {

count = 0;

}

COUNTING operator++() {//Operator Overloading

count = count+ 3;

return \*this;//using this->pointer to return count value

}

void display() {//display function to display value

cout << count;

}

};

int main()

{

COUNTING obj;

++obj; //increment the value of count to move one step a head

obj.display();//display function to display value

++obj;

cout << endl;

obj.display();

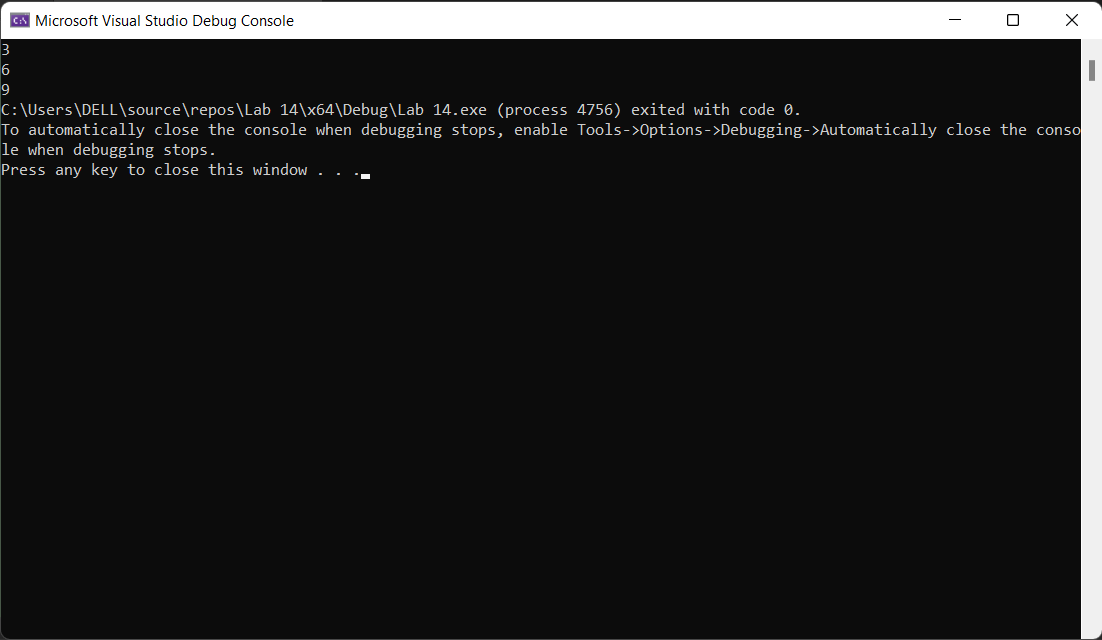
++obj;

cout << endl;

obj.display();

return 0;

}

****

**Task no 2:**

#include <iostream>

using namespace std;

class matrix {

public:

int arr[3][3];

int arr2[3][3];

int arr3[3][3];

matrix() {};

void setmatrix() {//The function sets the matrix input

cout << "Enter value of Matrix 3x3 One :" << endl;

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

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

cin >> arr[i][j];

}

}

cout << "Enter value of Matrix 3x3 Second :" << endl;

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

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

cin >> arr2[i][j];

}

}

}

matrix operator\*() {//this will use operator overloading where we multiple and add matrix

matrix mult;

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

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

mult.arr3[i][j] = 0;

for (int k = 0; k < 3; k++) {

mult.arr3[i][j] += arr[i][k] \* arr2[k][j];

}

}

}

return mult;

}

void display() {//function used to display the final result of amtrix

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

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

cout << arr3[i][j] << " ";

}

cout << endl;

}

}

};

int main()

{

matrix obj;

obj.setmatrix();

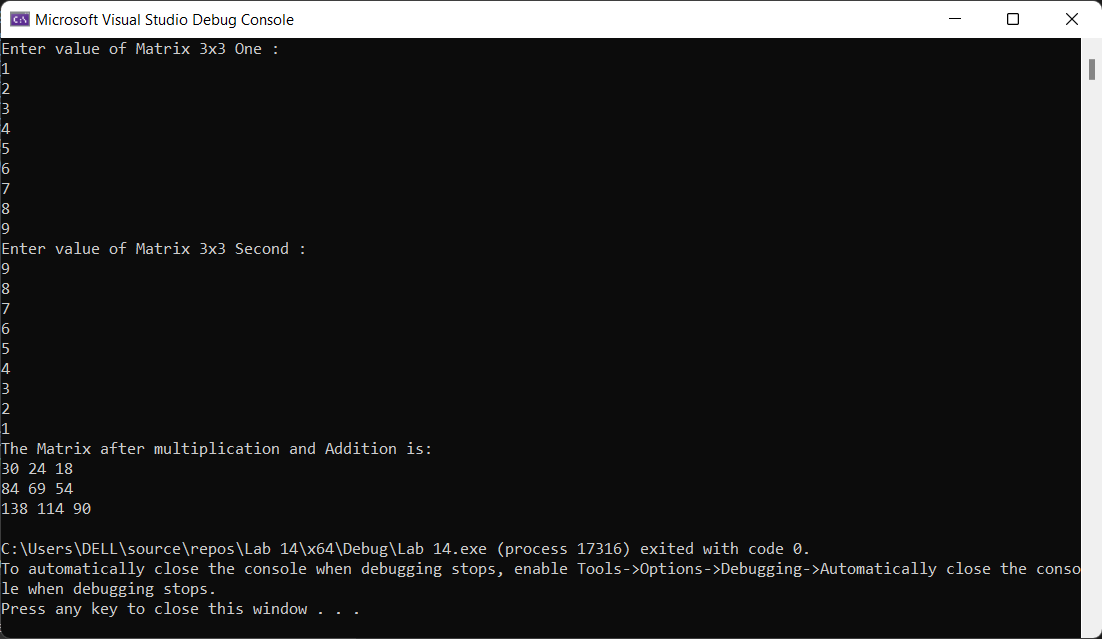
obj = \*obj;

cout << "The Matrix after multiplication and Addition is:"<<endl;

obj.display();

return 0;

}



**Task no 3:**

#include <iostream>

#include <string>

template <typename T>//using template to use it again and again

void swap(T& a, T& b) {//swap function

T temp = a;

a = b;

b = temp;

}

int main() {

int a = 5, b = 10;

std::cout << "Before swapping: a = " << a << ", b = " << b << std::endl;

swap(a, b);//swaping integer

std::cout << "After swapping: a = " << a << ", b = " << b << std::endl;

double c = 3.14, d = 2.71;

std::cout << "Before swapping: c = " << c << ", d = " << d << std::endl;

swap(c, d);//swaping double

std::cout << "After swapping: c = " << c << ", d = " << d << std::endl;

std::string e = "Hanan";

std:: string f = "Abdul" ;

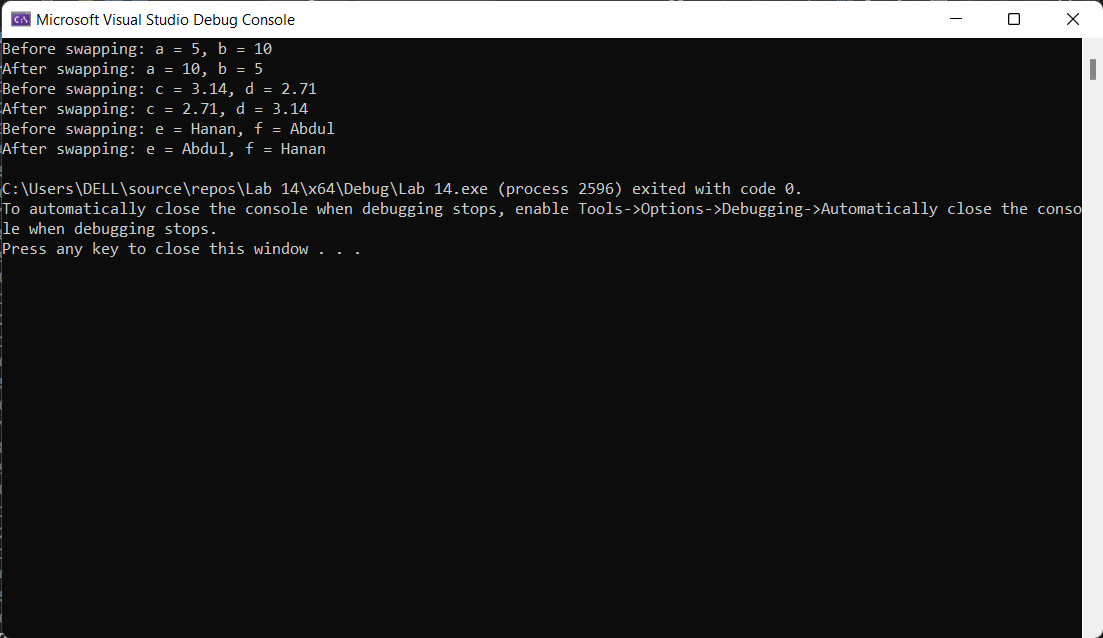
std::cout << "Before swapping: e = " << e << ", f = " << f << std::endl;

swap(e, f);//swaping string

std::cout << "After swapping: e = " << e << ", f = " << f << std::endl;

return 0;

}



**Task no 4**

#include <iostream>

template<typename T>

T greater(T& a, T& b, T& c, T& d) {//using template

if (a > b && a > c && a > d) {//checking conditions to check greater number

std::cout <<a<< " is greater :";

return a;

}

else if (b > a && b > c && b > d) {

std::cout <<b << " is greater :";

return b;

}

else if (c > a && c > b && c > d) {

std::cout <<c<< " is greater :";

return c;

}

else {

std::cout <<d<< " is greater :";

return d;

}

}

int main() {

std::cout << "Integer :" << std::endl;//using template for integer

int a = 3, b = 4, c = 5, d = 6;

std::cout<<greater(a, b, c, d);

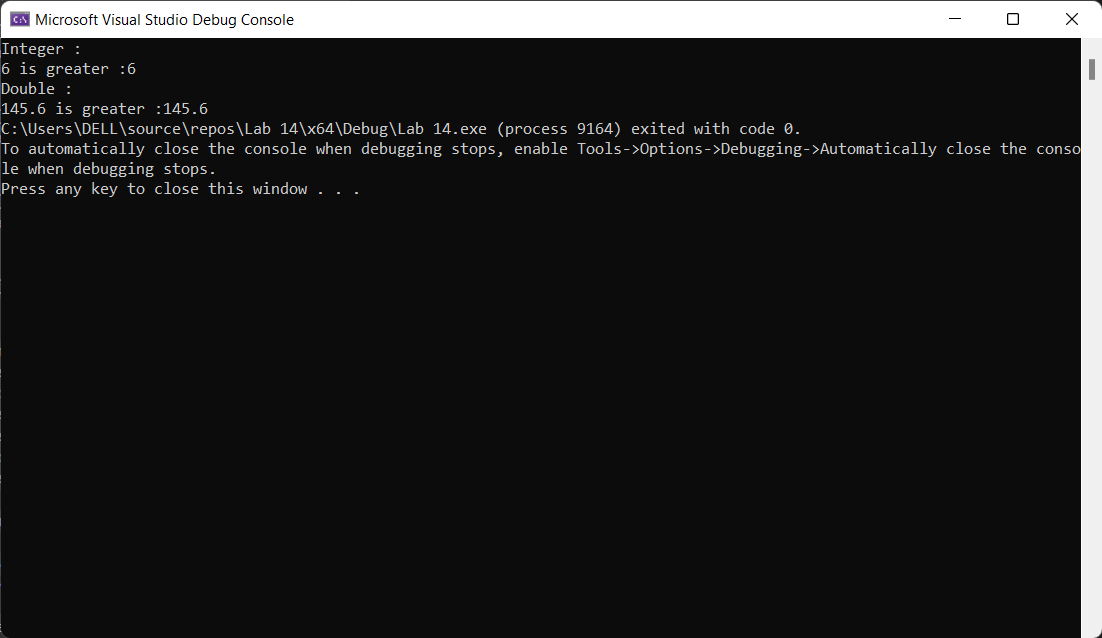
std::cout << "\nDouble :" << std::endl;

double e = 3.45, f = 55.4, g = 25.5, h = 145.6;//using template for double

std::cout << greater(e, f, g, h);

return 0;

}

****

**Task no 5:**

#include <iostream>

using namespace std;

template <class T>

class Number {

private:

T num1, num2;//varaible of Type T

public:

Number(T n1, T n2):num1(n1), num2(n2){}//constructor

T getnumber() { return num1; }

T getnumber2() { return num2; }

int division() {

cout << "Division of num1 and num2 is :";

return num1 / num2;

}

T getNum() {

return num1/num2;

}

};

int main() {

//create object with int type

Number<int>numberInt(7,2);

//create object with double type

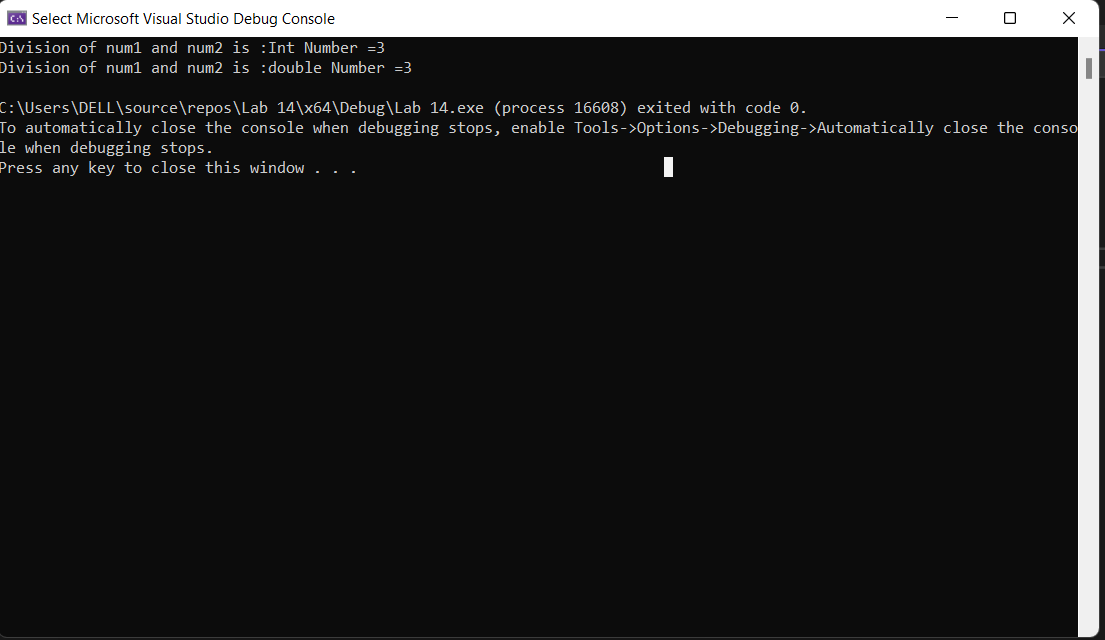
Number<double>numberDouble(7.7,3.8);

cout << "Int Number ="<< numberInt.division(); cout << endl;

cout << "double Number ="<< numberInt.division(); cout << endl;

return 0;

}

****