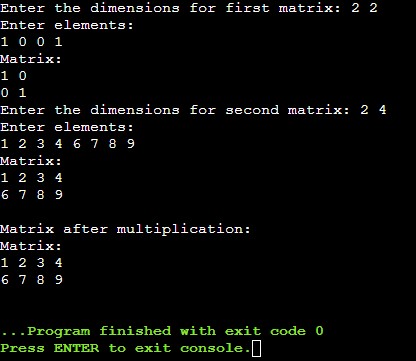
1. Write a C++ program for matrix multiplication with following specifications. a )Use constructor dynamic memory allocation for matrix b)Use getdata() function to input values for matrix c ) Use show( ) to display the matrix d)Use mul() to multiply two matrices:

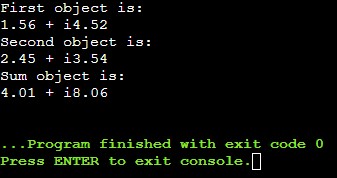
| #include <iostream> using namespace std;  class matrix{  int r, c, \*\*a;  public:  matrix(int row, int col) {  r = row;  c = col;  a = new int\*[r];  for(int i=0; i<c; i++)  a[i] = new int[c];  }  void getData() {  cout<<"Enter elements: "<<endl;  for(int i=0; i<r; i++)  for(int j=0; j<c; j++)  cin>>a[i][j];  }  void show() {  cout<<"Matrix: "<<endl;  for(int i=0; i<r; i++){  for(int j=0; j<c; j++)  cout<<a[i][j]<<" ";  cout<<endl;  }  }  void mul(matrix m2) {  matrix m1(r,m2.c);  int sum=0;  for(int i=0; i<r; i++){  for(int j=0; j<m2.c; j++){  for(int k=0; k<m2.r; k++)  sum = sum + a[i][k]\*m2.a[k][j];  m1.a[i][j] = sum;  sum = 0;  }  }  m1.show();  } };  int main() {  int x, y;  cout<<"Enter the dimensions for first matrix: ";  cin>>x>>y;  matrix m1(x,y);  m1.getData();  m1.show();  cout<<"Enter the dimensions for second matrix: ";  cin>>x>>y;  matrix m2(x,y);  m2.getData();  m2.show();  cout<<"\nMatrix after multiplication:"<<endl;  m1.mul(m2);   return 0; } |
| --- |



OUTPUT

4. Write a C++ program for addition of two complex numbers using friend function (use constructor function to initialize data members of complex class).

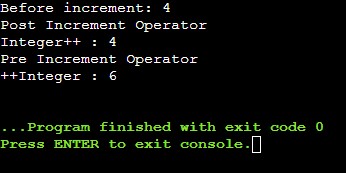
| #include <iostream> using namespace std;  class complex {  private:  float imag, real;  public:  complex(){  real = 0;  imag = 0;  }    complex(float x){  real = x;  imag = 0;  }    complex(float x, float y){  real = x;  imag = y;  }  friend void display(complex x1);  friend complex add(complex x1, complex x2); };  void display(complex x1){  cout<<x1.real<<" + i"<<x1.imag<<endl; }  complex add(complex x1, complex x2){  complex temp;  temp.real = x1.real + x2.real;  temp.imag = x1.imag + x2.imag;  return temp; }  int main() {  complex a(1.56, 4.52);  complex b(2.45, 3.54);  cout<<"First object is: "<<endl;  display(a);  cout<<"Second object is: "<<endl;  display(b);  complex c;  c = add(a,b);  cout<<"Sum object is: "<<endl;  display(c);  return 0; } |
| --- |



OUTPUT

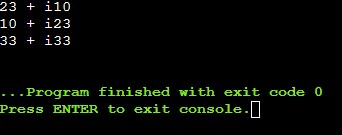
**Extra Question:** Operator overloading of pre-increment and post-increment operator

| #include <iostream>  using namespace std;  class test{  int a;  public:  test(int x=0){  this->a = x;  }  int getValue(){  return a;  }  test operator++(){  a++;  return \*this;  }  test operator++(int){  const test old(\*this);  ++(\*this);  return old;  }  };  int main()  {  test obj(4);  cout<<"Before increment: "<<obj.getValue()<<endl;  cout << "Post Increment Operator" << endl;  cout << "Integer++ : " << (obj++).getValue() << endl;  // cout << "Integer++ : " << (obj.operator++()).getValue() << endl;  cout << "Pre Increment Operator" << endl;  cout << "++Integer : " << (++obj).getValue() << endl;  return 0;  } |
| --- |



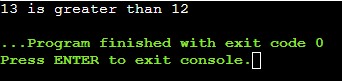
**Extra Question:** Complex addition

| #include <iostream> using namespace std;  class complex{  int real, imag;  public:  complex(int r=0, int c=0){  real = r;  imag = c;  }  void display(){  cout<<this->real<<" + i"<<this->imag<<endl;  }  complex operator+(complex c){  complex temp;  temp.real = this->real + c.real;  temp.imag = this->imag + c.imag;  return temp;  } };  int main() {  complex c1(23,10);  c1.display();  complex c2(10,23);  c2.display();  complex c3 = c1 + c2;  c3.display();   return 0; } |
| --- |



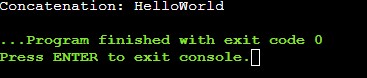
**Extra question:** relational operator

| #include <iostream> using namespace std;  class Integer{  int a;  public:  Integer(int i=0){  this->a = i;  }  void operator>(Integer obj){  if(this->a > obj.a)  cout<<this->a<<" is greater than "<<obj.a;  else  cout<<obj.a<<" is greater than "<<this->a;  } };  int main() {  Integer a(12), b(13);  a>b;  return 0; } |
| --- |



**Extra question:** Concatenate two strings

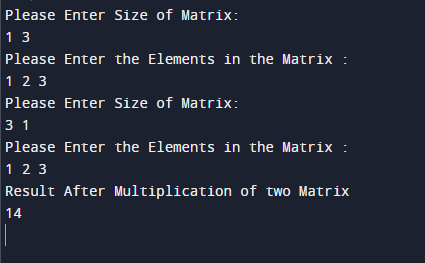
| #include <iostream> #include <string.h> using namespace std;   class AddString {   public:  char str[100];  AddString() {}    AddString(char str[]){  strcpy(this->str, str);  }    AddString operator+(AddString& S2) {  AddString S3;    strcat(this->str, S2.str);  strcpy(S3.str, this->str);  return S3;  } };   int main() {  char str1[] = "Hello";  char str2[] = "World";    AddString a1(str1);  AddString a2(str2);  AddString a3;  a3 = a1 + a2;  cout << "Concatenation: " << a3.str;    return 0; } |
| --- |



1. Modify the above program as follows

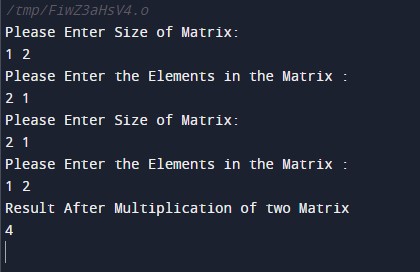
a) Use operator\*() for matrix multiplication instead of mul( )

| #include<iostream> using namespace std; class matrix {  private:  int a[10][10];  int u,v;  public:  matrix operator \*(matrix);  void show();  void read(); };  //\* operator is overloaded  matrix matrix::operator\*(matrix m2) {  matrix t;  t.u=u;  t.v=m2.v;  for(int i=0;i<t.u;i++)  for(int j=0;j<t.v;j++){  t.a[i][i]=0;  for(int k=0;k<v;k++)  t.a[i][j] += a[i][k] \* m2.a[k][j];  }  return t; }   void matrix::read(){  cout<<"Please Enter Size of Matrix:\n";  cin>>u>>v;  cout<<"Please Enter the Elements in the Matrix :\n";  for(int i=0;i<u;i++)  for(int j=0;j<v;j++)  cin>>a[i][j]; }  void matrix::show() {  for(int i=0;i<u;i++) {  for(int j=0;j<v;j++)  cout<<a[i][j]<<"\t";  cout<<"\n";  } }  int main() {  matrix object1, object2, object3;  object1.read();  object2.read();  object3 = object1 \*object2;  cout<<"Result After Multiplication of two Matrix\n";  object3.show(); } |
| --- |



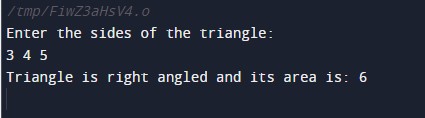
b)Make operator\*() as friend function

| #include<iostream> using namespace std; class matrix {  private:  int a[10][10];  int u,v;  public:  friend matrix operator \*(matrix, matrix);  void show();  void read(); };  //\* operator is overloaded matrix operator\*(matrix m1, matrix m2) {  matrix t;  t.u = m1.u;  t.v = m2.v;  for(int i=0;i<t.u;i++)  for(int j=0;j<t.v;j++){  t.a[i][i]=0;  for(int k=0;k<m1.v;k++)  t.a[i][j] += m1.a[i][k] \* m2.a[k][j];  }  return t; }   void matrix::read(){  cout<<"Please Enter Size of Matrix:\n";  cin>>u>>v;  cout<<"Please Enter the Elements in the Matrix :\n";  for(int i=0;i<u;i++)  for(int j=0;j<v;j++)  cin>>a[i][j]; }  void matrix::show() {  for(int i=0;i<u;i++) {  for(int j=0;j<v;j++)  cout<<a[i][j]<<"\t";  cout<<"\n";  } }  int main() {  matrix object1, object2, object3;  object1.read();  object2.read();  object3 = operator\*(object1, object2);  cout<<"Result After Multiplication of two Matrix\n";  object3.show(); } |
| --- |



1. Write a C++ program to compute the area of right angle triangle, equilateral triangle, isosceles triangle using function overloading concept.

| #include<iostream> #include<cmath> using namespace std;  class Triangle{  float a, b, c;  public:  Triangle(float s1, float s2, float s3){  a = s1;  b = s2;  c = s3;  check();  }    void check(){  if((a+b)>c && (b+c)>a && (c+a)>b){  if(a==b && b==c)  area(a);  else if((a\*a+b\*b-c\*c == 0) || (c\*c+b\*b-a\*a == 0) || (a\*a+c\*c-b\*b == 0)){  if(a\*a+b\*b-c\*c == 0)  area(a,b);  else if(c\*c+b\*b-a\*a == 0)  area(c,b);  else  area(a,c);  }  else if(a==b || b==c || c==a)  area(a,b,c);  else  area(a,b,c);  }  else{  cout<<"Triangle not possible!!"<<endl;  return;  }  }    void area(float a){  float area = (pow(3,0.5)/4)\*pow(a,2);  cout<<"Triangle is equilateral and its area is: "<<area<<endl;  }    void area(float a, float b){  float area = 0.5 \* a \*b;  cout<<"Triangle is right angled and its area is: "<<area<<endl;  }    void area(float a, float b, float c){  float s = (a + b + c)/2;  float area, x;  x = s \* (s-a) \* (s-b) \* (s-c);  area = pow(x,0.5);  if(a==b || b==c || c==a)  cout<<"Triangle is isosceles and its area is: "<<area<<endl;  else  cout<<"Triangle is scalene and its area is: "<<area<<endl;  } };  int main(){  float x1,y1,z1;  cout<<"Enter the sides of the triangle: "<<endl;  cin>>x1>>y1>>z1;  Triangle a(x1,y1,z1);  return 0; } |
| --- |



5. Define a class string and overload = = to compare two strings and + operator for concatenation of two strings

| #include<iostream> #include<cstring> using namespace std;  class compare { private:  char a[30]; public:   void getdata(){  cout<<"Enter the string: "<<endl;  cin>>a;  }  void display(){  cout<<"String is: "<<a<<endl;  }   compare operator+(compare s1){  compare temp;  strcpy(temp.a,this->a);  strcat(temp.a,s1.a);  return temp;  }   int operator==(compare s1){  if(strcmp(this->a,s1.a)==0)  return 1;  else  return 0;  } };  int main() {  compare a,b,c;  a.getdata();  b.getdata();  c=a+b;  c.display();  if(a==b)  cout<<"Equal strings"<<endl;  else  cout<<"Not equal strings"<<endl; } |
| --- |

