**INDEX**

Submitted by: Aditya Singh

Roll no: A2305220142

Section: 3CSE-2Y

| **S. No** | **Content** | **Date** |
| --- | --- | --- |
| **1.** | Write a program to check palindrome for a) string b) number | 16/8/21 |
| **2.** | Write a program to show effect of Call by value and Call by Reference | 16/8/21 |
| **3.** | Write a program to perform matrix algebra operations: Addition, Subtraction, Multiplication, Transpose. | 16/8/21 |
| **4** | Write a program to read a matrix of size m x n from the keyboard and display the same on the screen using functions. | 16/8/21 |
| **5.** | Rewrite above program, to make row parameter of the matrix as a default argument. | 16/8/21 |
| **6.** | The effect of a default argument can be alternatively achieved by overloading. Discuss with an example. | 16/8/21 |
| **7.** | Write a macro that obtains the largest of three numbers. | 16/8/21 |
| **8.** | Redo above program using inline function. | 16/8/21 |
| **9.** | Write a function power () to raise a number m to a power n. The function takes a double value for m and int galue for n, and returns the result correctly. Use a default value of 2 for n to make the function to calculate squares when this argument is omitted. | 16/8/21 |
| **10.** | Write a function that performs the same operation as that of above question but takes an int value for m. Both the functions should have the same name. Write a main that calls both the function. | 16/8/21 |
| **11.** | Create a class date having day, month, and year. Use the concept of constructor overloading to initialize the value. Create object of class Date. | 6/9/21 |
| **12.** | Take time of two zones, current time of city 1 and current time of city 2. Calculate the time difference between two cities. Use the concept of passing objects as parameters. | 6/9/21 |
| **13.** | Write a program to demonstrate simple inheritance. | 17/9/21 |
| **14.** | Write a program to demonstrate private simple inheritance. | 17/9/21 |
| **15.** | Write a program to demonstrate protected simple inheritance. | 17/9/21 |
| **16.** | Write a program to read and print student’s information using two classes and simple inheritance. | 17/9/21 |
| **17.** | Write a program to demonstrate multilevel inheritance. | 17/9/21 |
| **18.** | Write a program to read and print employee information using multiple inheritance. | 17/9/21 |
| **19.** | Write a program to demonstrate multiple inheritance | 17/9/21 |
| **20.** | Write a program to demonstrate hierarchical inheritance to get square and cube of a number. | 17/9/21 |
| **21.** | Write a program to read and print employee information with department and PF information using hierarchical inheritance. | 17/9/21 |
| **22.** | Design three classes: Student, Exam and Result. The student class has data members such as roll no, name etc. Create a class Exam by inheriting the student class. The Exam class adds data members representing the marks scored in six subjects. Derive the Result from class Exam and it has its own members such as total marks. Write an interactive program to model this relationship. | 8/10/21 |
| **23.** | Consider an example of book shop which sells books and video tapes. These two classes are inherited from base class called media. The media class has command data members such as title and publication. The book class has data members for storing number of pages in a book and tape class has playing time in a tape, Each class will have member functions such as read() and show(). In the base class, these members have to be defined as virtual functions. Write a program to models the class the class hierarchy for book shop and processes objects of these classes using pointers to base class. | 8/10/21 |
| **24.** | Write a program to demonstrate operator overloading: Addition of two Complex Numbers. | 22/10/21 |
| **25.** | Write a program to demonstrate operator overloading: Evaluating Fractions. | 22/10/21 |
| **26.** | Write a program to demonstrate an example of virtual functions: Displaying derived and base class virtual functions. | 29/10/21 |
| **27.** | Write a program to demonstrate an example of virtual functions: Calculating area of circle and cylinder. | 29/10/21 |
| **28.** | Write a program to demonstrate file handling: Creating a binary file to store student details. | 12/11/21 |
| **29.** | Write a program to demonstrate Exception handling: | 12/11/21 |
| **30.** | Write a program to demonstrate Exception handling: | 12/11/21 |

**1. Aim**: Write a program to check palindrome for a) string b) number

**Program:**

#include<iostream>

#include<cstring>

void int\_palindrome()

{

int n,d,n1=0,temp,i;

std::cout<<"\nEnter an integer: ";

std::cin>>n;

temp=n;

while(temp!=0)

{

d=temp%10;

n1=(n1\*10)+d;

temp=temp/10;

}

if(n==n1)

std::cout<<"Given number is a palindrome.";

else

std::cout<<"Number is not a palindrome.";

}

void str\_palindrome()

{

int len,i,f=0;

char str[50];

std::cout<<"\n\nEnter a string: ";

std::cin>>str;

len=strlen(str);

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

{ if(str[i] != str[len-i-1])

{

f=1;

break;

}

}

if(f==1)

std::cout<<"Not a palindrome";

else

std::cout<<"It is a palindrome";

}

int main()

{

int ch;

char c='y';

do{

std::cout<<"Enter your choice: ";

std::cout<<"\n1.Check whether string is palindrome\n2.Check whether given integer is a paldnrome.\nInput: ";

std::cin>>ch;

switch(ch)

{

case 1: str\_palindrome();

break;

case 2: int\_palindrome();

break;

default: std::cout<<"Invalid option.";

break;

}

std::cout<<"\n\nRepeat?(y/n)";

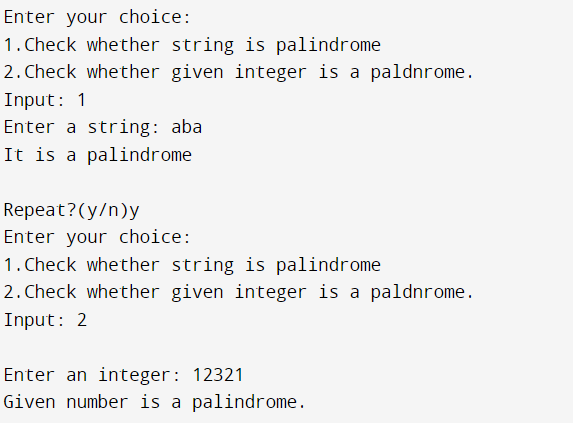
std::cin>>c;

}while(c=='Y'||c=='y');

return 0;

}

**Output:**



**2. Aim**: Write a program to show effect of Call by value and Call by Reference

**Program**:

#include <iostream>

void swapbyval(int a,int b)

{

int t;

t=a;a=b;b=t;

std::cout<<"\nValue of a= "<<a<<"\nValue of b= "<<b;

std::cout<<"\n\nAddress of a="<<&a<<"\nAddress of b= "<<&b;

}

void swapbyref(int &a,int &b)

{

int t;

t=a;a=b;b=t;

std::cout<<"\nValue of a= "<<a<<"\nValue of b= "<<b;

std::cout<<"\n\nAddress of a="<<&a<<"\nAddress of b= "<<&b;

}

int main()

{

int a,b,ch;

std::cout<<"\nEnter two numbers a and b: ";

std::cin>>a>>b;

std::cout<<"\n\nAddress of a="<<&a<<"\nAddress of b= "<<&b;

std::cout<<"\n1.Display call by value.\n2.Display call by reference.\nChoice: ";

std::cin>>ch;

switch(ch)

{

case 1: swapbyval(a,b);

break;

case 2: swapbyref(a,b);

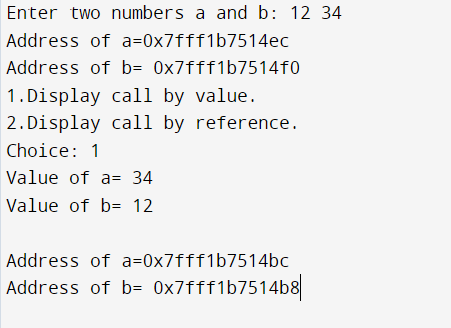
break;

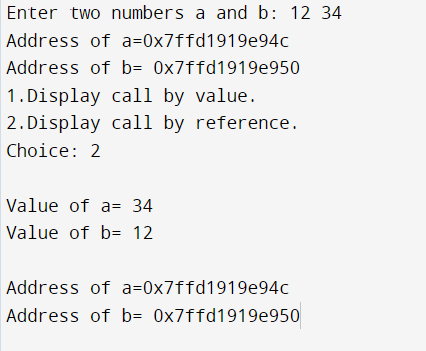
}

return 0;

}

**Output:**





**3. Aim:** Write a program to perform matrix algebra operations: Addition, Subtraction, Multiplication, Transpose.

**Program:**

#include<iostream>

int add()

{

int i,j,r,c,r2,c2,a[10][10],b[10][10],sum[10][10];

std::cout<<"\n\nEnter number of rows in M1: ";

std::cin>>r;

std::cout<<"Enter number of columns in C1: ";

std::cin>>c;

std::cout<<"\n\nEnter number of rows in M2: ";

std::cin>>r2;

std::cout<<"Enter number of columns in C2: ";

std::cin>>c2;

if(r==r2&&c==c2)

{

std::cout<<"\nAddition possible";

std::cout<<"\n\nEnter elements in M1 "<<r<<"\*"<< c<<"\n";

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

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

std::cin>>a[i][j];

std::cout<<"\n\nEnter elements in M2:\n";

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

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

std::cin>>b[i][j];

std::cout<<"Sum=\n";

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

{

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

{

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

std::cout<<"| "<<sum[i][j]<<" | ";

}

std::cout<<"\n";

}

}

else{

std::cout<<"Addition not possible";

exit(0);}

return 0;

}

int subtract()

{

int i,j,r,c,r2,c2,a[10][10],b[10][10],diff[10][10];

std::cout<<"\nEnter number of rows in M1: ";

std::cin>>r;

std::cout<<"\nEnter number of columns in C1: ";

std::cin>>c;

std::cout<<"\nEnter number of rows in M2: ";

std::cin>>r2;

std::cout<<"\nEnter number of columns in C2: ";

std::cin>>c2;

if(r==r2&&c==c2)

{ std::cout<<"\nSubtraction possible";

std::cout<<"\n\nEnter elements in M1 "<<r<<"\*"<< c<<"\n";

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

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

std::cin>>a[i][j];

std::cout<<"\nEnter elements in M2\n";

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

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

std::cin>>b[i][j];

std::cout<<"Difference:\n";

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

{

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

{

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

std::cout<<diff[i][j]<<" | ";

}

std::cout<<"\n";

}

}

else

{

std::cout<<"Subtraction not possible.\n";

exit(0);}

return 0;

}

int multiply()

{

int i,j,k,r,c,r2,c2,a[10][10],b[10][10],mult[10][10];

std::cout<<"\nEnter number of rows in M1: ";

std::cin>>r;

std::cout<<"\nEnter number of columns in C1: ";

std::cin>>c;

std::cout<<"\nEnter number of rows in M2: ";

std::cin>>r2;

std::cout<<"\nEnter number of columns in C2: ";

std::cin>>c2;

if(r2==c)

{

std::cout<<"Multiplication possible\n";

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

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

mult[i][j]=0;

std::cout<<"\n\nEnter elements in M1 "<<r<<"\*"<< c<<"\n";

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

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

std::cin>>a[i][j];

std::cout<<"\nEnter elements in M2\n";

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

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

std::cin>>b[i][j];

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

for(j=0;j<c2;j++)

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

{

mult[i][j]+=a[i][k]\*b[k][j];

}

for(i=0;i<r;i++){

for(j=0;j<c2;j++)

{

std::cout<<"| "<<mult[i][j]<<" |";

}std::cout<<"\n";

}

}

else

{

std::cout<<"Multiplication not possible\n";

}

return 0;

}

int transpose()

{

int a[10][10],r,c,i,j;

std::cout<<"\nEnter number of rows in Matrix: ";

std::cin>>r;

std::cout<<"\nEnter number of columns in Matrix: ";

std::cin>>c;

std::cout<<"\n\nEnter elements in Matrix:\n";

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

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

std::cin>>a[i][j];

std::cout<<"\nMatrix input:\n";

for(i=0;i<r;i++){

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

{

std::cout<<"| "<<a[i][j]<<" |";

}std::cout<<"\n";

}

std::cout<<"Transpose:\n";

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

{

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

std::cout<<"| "<<a[j][i]<<" |";

std::cout<<"\n";

}

return 0;

}

int main()

{

int ch;

std::cout<<"Enter your choice: \n1.Add two matrices\n2.Subtract two matrices\n3.Multiply two matrices\n4.Print transpose of a matrix.";

std::cout<<"\nChoice: ";

std::cin>>ch;

switch(ch)

{

case 1: add();

break;

case 2: subtract();

break;

case 3: multiply();

break;

case 4: transpose();

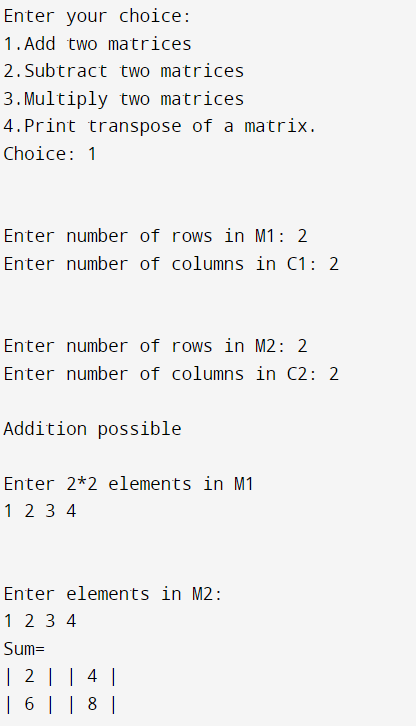
break;

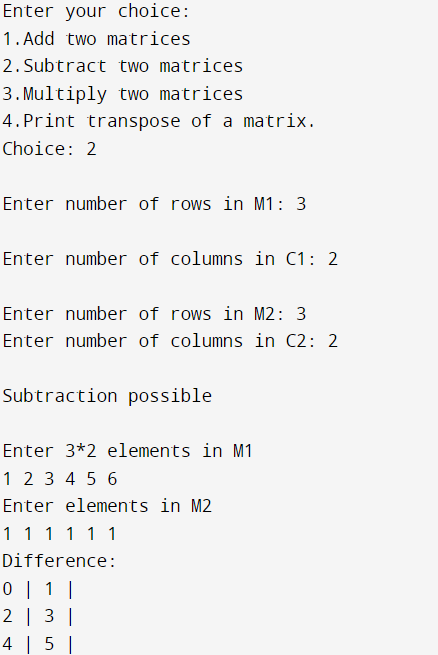
}

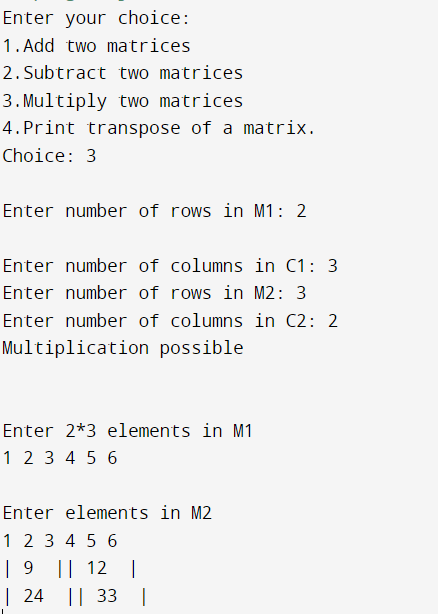
return 0;

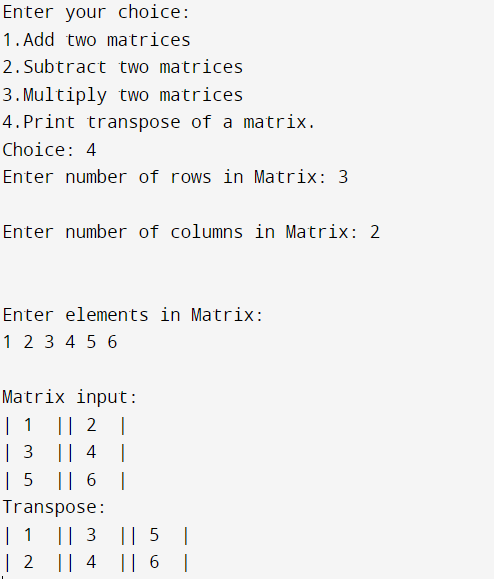
}

**Output:**









**4. Aim**: Program to read a matrix of size m x n from the keyboard, and display the same on the screen using functions.

**Program:**

#include <iostream>

int a[10][10];

int inp(int m,int n)

{

int i,j;

std::cout<<"Enter "<<m<<"\*"<<n<<" elements:\n";

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

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

std::cin>>a[i][j];

return 0;

}

int disp(int m,int n)

{

int i,j;

std::cout<<"Matrix elements:\n";

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

{

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

std::cout<<"| "<<a[i][j]<<" |";

std::cout<<"\n";

}

return 0;

}

int main() {

int m,n;

std::cout<<"Enter number of rows in matrix: ";

std::cin>>m;

std::cout<<"\nEnter number of columns in matrix: ";

std::cin>>n;

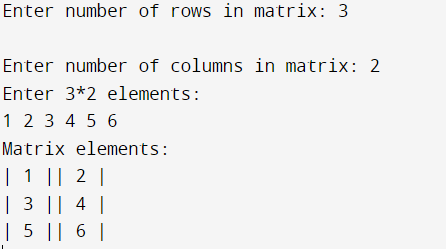
inp(m,n);

disp(m,n);

return 0;

}

**Output:**



**5. Aim**: Rewrite above program, to make row parameter of the matrix as a default argument.

**Program:**

#include <iostream>

int a[10][10];

int inp(int n,int m=3)

{

int i,j;

std::cout<<"Enter "<<n<<"\*"<<m<<" elements:\n";

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

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

std::cin>>a[i][j];

return 0;

}

int disp(int n,int m=3)

{

int i,j;

std::cout<<"Matrix elements:\n";

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

{

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

std::cout<<"| "<<a[i][j]<<" |";

std::cout<<"\n";

}

return 0;

}

int main() {

int n;

std::cout<<"\nEnter number of columns in matrix: ";

std::cin>>n;

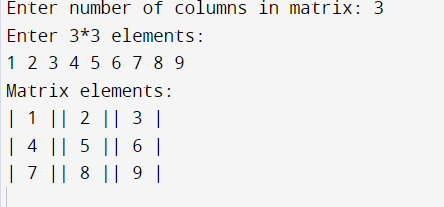
inp(n);

disp(n);

return 0;

}

**Output:**



**6. Aim:** The effect of a default argument can be alternatively achieved by overloading. Discuss with an example.

**Program**:

#include <iostream>

int sum(int a,int b)

{

std::cout<<(a+b)<<"\n";

}

int sum(int a,int b,int c)

{

std::cout<<(a+b+c)<<"\n";

}

int main()

{

int a,b,c;

std::cout<<"Enter three numbers: ";

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

sum(a,b,c);

std::cout<<"Enter two numbers: ";

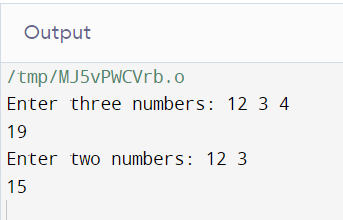
std::cin>>a>>b;

sum(a,b);

return 0;

}

**Output:**



**7. Aim**: Write a program with a macro that obtains the largest of three numbers.

**Program**:

#include<iostream>

#define MAX(a,b,c)(a>b?(a>c?a:c):(b>c?b:c))

int main()

{

int a,b,c,d;

std::cout<<"Enter three integers:\n";

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

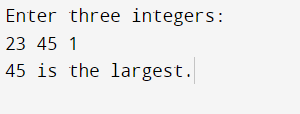
d=MAX(a,b,c);

std::cout<<d<<" is the largest.";

return 0;

}

**Output**:



**8. Aim**: Redo above program using inline function.

**Program:**

#include <iostream>

inline int MAX(int a,int b,int c)

{

int big;

big=a>b?(a>c?a:c):(b>c?b:c);

return big;

}

int main() {

// Write C++ code here

int a,b,c,big;

std::cout<<"Enter three numbers: ";

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

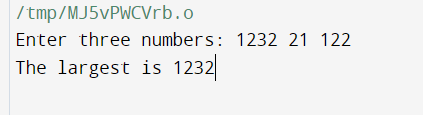
big=MAX(a,b,c);

std::cout<<"The largest is "<<big;

return 0;

}

**Output**:



**9. Aim:** Write a function power() to raise a number m to a power n. The function takes a double value for m and int galue for n, and returns the result correctly. Use a default value of 2 for n to make the function to calculate squares when this argument is omitted.

**Program:**

#include<iostream>

int power(double m,int n=2)

{

int product=1;

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

product\*=m;

return product;

}

int main()

{

double m;

int n,p;

std::cout<<"Enter m: ";

std::cin>>m;

std::cout<<"Enter n: ";

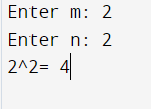
std::cin>>n;

p=power(m,n);

std::cout<<m<<"^"<<n<<"= "<<p;

}

**Output**:



**10. Aim**: Write a function that performs the same operation as that of above question but takes an int value for m. Both the functions should have the same name. Write a main that calls both the function.

**Program:**

#include <iostream>

float power(double m,int n)

{

float product=1.0;

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

product\*=m;

return product;

}

int power(int m,int n)

{

int product=1;

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

product\*=m;

return product;

}

int main()

{

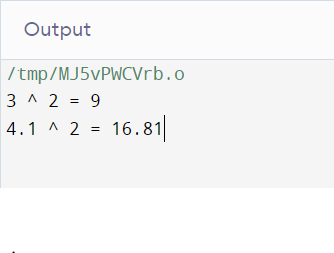
std::cout<<"3 ^ 2 = "<<power(3,2);

std::cout<<"\n4.1 ^ 2 = "<<power(4.1,2);

return 0;

}

**Output**:



**11. Aim:** Create a class date having day, month, and year. Use the concept of constructor overloading to initialize the value. Create object of class Date.

**Program:**

#include <iostream>

using namespace std;

class Date

{

int dd,mm,yy;

public:

void input()

{

cout<<"Enter day: ";

cin>>dd;

cout<<"Enter month: ";

cin>>mm;

cout<<"Enter Year: ";

cin>>yy;

}

void display()

{

cout<<"\nDay: "<<dd;

cout<<"\nMonth: "<<mm;

cout<<"\nYear: "<<yy;

}

Date()

{

dd=1,mm=1,yy=1;

}

Date(int d,int m,int y)

{

dd=d,mm=m,yy=y;

}

};

int main()

{

Date d;

Date d1(12,12,12);

cout<<"\nConstructer call";

d.display();

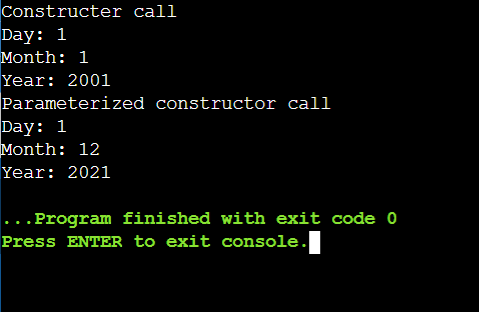
cout<<"\nParameterized constructor call";

d1.display();

return 0;

}

**Output:**



**12. Aim:** Take time of two zones, current time of city 1 and current time of city 2. Calculate the time difference between two cities. Use the concept of passing objects as parameters.

**Program:**

#include <iostream>

using namespace std;

class Time

{

int hh, mm; //ss;

int hdiff,mdiff;

public:

void input()

{

cout << "Enter hours: ";

cin >> hh;

cout << "Enter minutes: ";

cin >> mm;

}

void display()

{

cout << "Hours: " << hh;

cout << "Minutes: " << mm;

}

void diff(Time t1,Time t2)

{

if(t2.mm>t1.mm)

{

t1.hh-=1;

t1.mm+=60;

}

hdiff=t1.hh-t2.hh;

mdiff=t1.mm-t2.mm;

cout<<"\nDifference in hours: "<<hdiff;

cout<<"\nDifference in minutes: "<<mdiff;

}

void convertT(Time t1,Time t2)

{

int th1,th2,mh1,mh2;

th1=t1.hh+hdiff;

cout<<"Converted Time in city 1";

cout<<"Hours "<<th1;

//cout<<"Minutes: "<<t1.mm+mdiff;

}

};

int main()

{

Time t1, t2, t3;

cout<<"\nInput time in city 1\n";

t1.input();

cout<<"\nInput time in City 2\n";

t2.input();

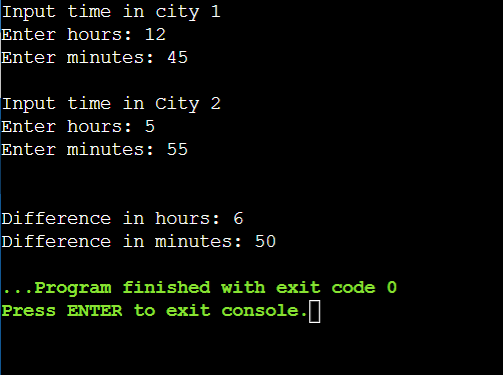
cout<<"\n";

t3.diff(t1,t2);

return 0;

}

**Output:**



**13. Aim:** Write a program to demonstrate simple inheritance.

**Program:**

#include <iostream>

using namespace std;

class parent

{

char f\_name[30],m\_name[30];

public:

void pinput()

{

cout<<"\nEnter father's name: ";

cin>>f\_name;

cout<<"\nEnter mother's name: ";

cin>>m\_name;

}

void p\_output()

{

cout<<"\nFather's name: ";

cout<<f\_name;

cout<<"\nMother's name: ";

cout<<m\_name;

}

};

class child:public parent

{

char c\_name[30];

public:

void cinput()

{

cout<<"\nEnter child's name: ";

cin>>c\_name;

}

void c\_output()

{

cout<<"\nChild's name: ";

cout<<c\_name;

}

};

int main()

{

child c;

c.cinput();

c.pinput();

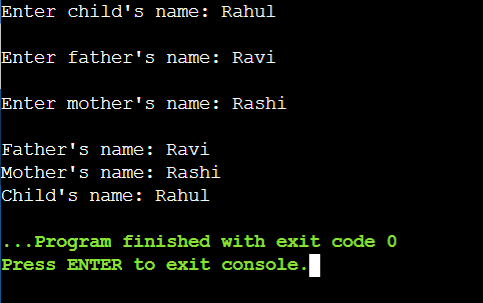
c.p\_output();

c.c\_output();

return 0;

}

**Output:**



**14. Aim:** Write a program to demonstrate private simple inheritance.

**Program:**

#include <iostream>

using namespace std;

class parent

{

char f\_name[30],m\_name[30];

public:

void pinput()

{

cout<<"\nEnter father's name: ";

cin>>f\_name;

cout<<"\nEnter mother's name: ";

cin>>m\_name;

}

void p\_output()

{

cout<<"\nFather's name: ";

cout<<f\_name;

cout<<"\nMother's name: ";

cout<<m\_name;

}

};

class child:private parent

{

char c\_name[30];

public:

void cinput()

{

cout<<"\nEnter child's name: ";

cin>>c\_name;

pinput();

}

void c\_output()

{

cout<<"\nChild's name: ";

cout<<c\_name;

p\_output();

}

};

int main()

{

child c;

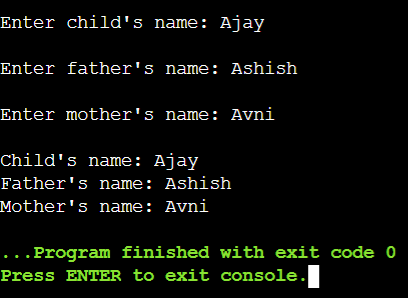
c.cinput();

c.c\_output();

return 0;

}

**Output:**



**15. Aim:** Write a program to demonstrate protected simple inheritance.

**Program:**

#include <iostream>

using namespace std;

class parent

{

protected:

char f\_name[30],m\_name[30];

public:

void pinput()

{

cout<<"\nEnter father's name: ";

cin>>f\_name;

cout<<"\nEnter mother's name: ";

cin>>m\_name;

}

};

class child:protected parent

{

char c\_name[30];

public:

void cinput()

{

cout<<"\nEnter child's name: ";

cin>>c\_name;

pinput();

}

void c\_output()

{

cout<<"\nChild's name: ";

cout<<c\_name;

cout<<"\nFather's name: ";

cout<<f\_name;

cout<<"\nMother's name: ";

cout<<m\_name;

}

};

int main()

{

child c;

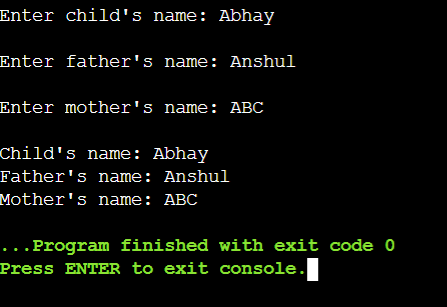
c.cinput();

c.c\_output();

return 0;

}

**Output:**



**16. Aim:** Write a program to read and print student’s information using two classes and simple inheritance.

**Program:**

#include <iostream>

using namespace std;

class parent

{

char f\_name[30],m\_name[30];

public:

void pinput()

{

cout<<"Enter father's name: ";

cin>>f\_name;

cout<<"Enter mother's name: ";

cin>>m\_name;

}

void p\_output()

{

cout<<"\nFather's name: ";

cout<<f\_name;

cout<<"\nMother's name: ";

cout<<m\_name;

}

};

class student:public parent

{

int admn\_no;

char c\_name[30];

public:

void sinput()

{

cout<<"Enter student's name: ";

cin>>c\_name;

cout<<"Enter admission number: ";

cin>>admn\_no;

}

void s\_output()

{

cout<<"\nStudent's name: ";

cout<<c\_name;

cout<<"\nAdmission number: ";

cout<<admn\_no;

}

};

int main()

{

student s;

s.sinput();

s.pinput();

cout<<"\n--Student Details--";

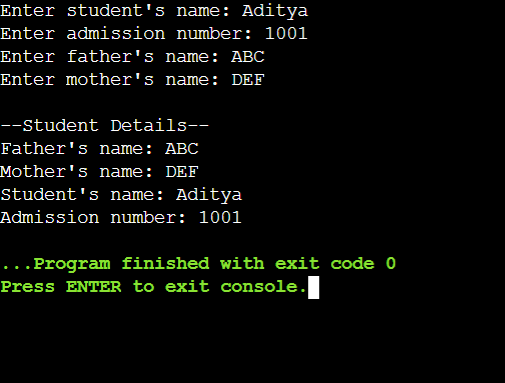
s.p\_output();

s.s\_output();

return 0;

}

**Output:**



**17. Aim:** Write a program to demonstrate multilevel inheritance.

**Program:**

#include <iostream>

using namespace std;

class student

{

protected:

int roll;

public:

void r\_input()

{

cout<<"\nEnter Roll number: ";

cin>>roll;

}

void r\_output()

{

cout<<"\nRoll number: "<<roll;

}

};

class test:public student

{

protected:

int marks[2];

public:

void m\_input()

{

int i;

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

{

cout<<"\nEnter marks in Subject "<<i+1<<": ";

cin>>marks[i];

}

}

void m\_output()

{

int i;

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

{

cout<<"\nMarks in Subject "<<i+1<<": ";

cout<<marks[i];

}

}

};

class result:public test

{

private:

float total,average;

public:

void res()

{

cout<<"\nReport Card:\n";

r\_output();

m\_output();

total=marks[0]+marks[1];

average=total/2;

cout<<"\nTotal: "<<total;

cout<<"\nAverage: "<<average;

}

};

int main()

{

result s;

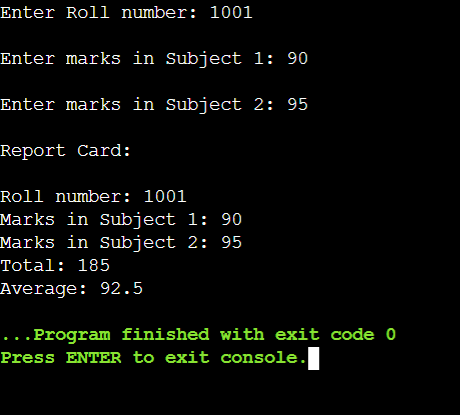
s.r\_input();

s.m\_input();

s.res();

}

**Output:**



**18. Aim:** Write a program to read and print employee information using multiple inheritance.

**Program:**

#include <iostream>

using namespace std;

class emp\_details

{

protected:

int emp\_no;

char phn\_no[10],emp\_name[30];

public:

void input()

{

cout<<"\nEnter Employee's name: ";

cin>>emp\_name;

cout<<"\nEnter Employee number: ";

cin>>emp\_no;

cout<<"\nEnter phone number: ";

cin>>phn\_no;

}

void ouput()

{

cout<<"\nEmployee's name: "<<emp\_name;

cout<<"\nEmployee's number: "<<emp\_no;

cout<<"\nEmployee phone number: "<<phn\_no;

}

};

class salary

{

protected:

int basic,da,hra;

public:

void b\_input()

{

cout<<"\nEnter basic: ";

cin>>basic;

cout<<"\nEnter HRA: ";

cin>>hra;

cout<<"\nEnter DA: ";

cin>>da;

}

};

class details:public emp\_details,public salary

{

float salary;

public:

void calc\_salary()

{

salary=basic+(hra\*basic)/100+(da\*basic)/100;

cout<<"\nGross Salary: "<<salary;

}

};

int main()

{

details d;

d.input();

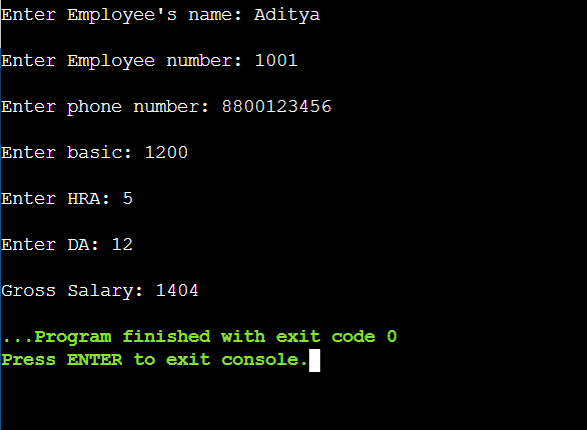
d.b\_input();

d.calc\_salary();

return 0;

}

**Output:**



**19. Aim:** Write a program to demonstrate multiple inheritance

**Program:**

#include <iostream>

using namespace std;

class emp\_details

{

protected:

int emp\_no;

char phn\_no[10],emp\_name[30];

public:

void input()

{

cout<<"\nEnter Employee's name: ";

cin>>emp\_name;

cout<<"\nEnter Employee number: ";

cin>>emp\_no;

cout<<"\nEnter phone number: ";

cin>>phn\_no;

}

void ouput()

{

cout<<"\nEmployee's name: "<<emp\_name;

cout<<"\nEmployee's number: "<<emp\_no;

cout<<"\nEmployee phone number: "<<phn\_no;

}

};

class salary

{

protected:

int basic,da,hra;

public:

void b\_input()

{

cout<<"\nEnter basic: ";

cin>>basic;

cout<<"\nEnter HRA: ";

cin>>hra;

cout<<"\nEnter DA: ";

cin>>da;

}

};

class details:public emp\_details,public salary

{

float salary;

public:

void calc\_salary()

{

salary=basic+(hra\*basic)/100+(da\*basic)/100;

cout<<"\nGross Salary: "<<salary;

}

};

int main()

{

details d;

d.input();

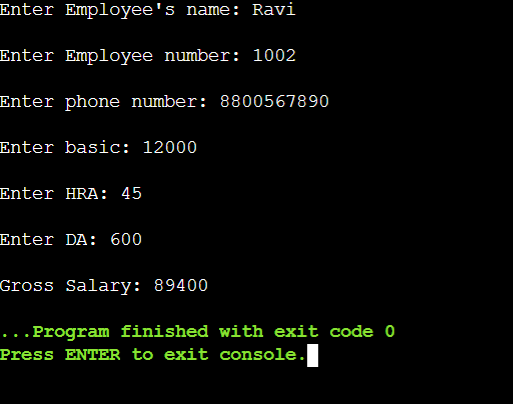
d.b\_input();

d.calc\_salary();

return 0;

}

**Output:**



**20. Aim:** Write a program to demonstrate hierarchical inheritance to get square and cube of a number.

**Program:**

#include <iostream>

using namespace std;

class num

{

protected:

int no;

public:

void input()

{

cout<<"\n\nEnter number: ";

cin>>no;

}

};

class square:public num

{

int square;

public:

void sq\_calc()

{

input();

square=no\*no;

cout<<"\nSquare: "<<square;

}

};

class cube:public num

{

public:

void c\_calc()

{

input();

cout<<"\nCube: "<<(no\*no\*no);

}

};

int main()

{

square s;

s.sq\_calc();

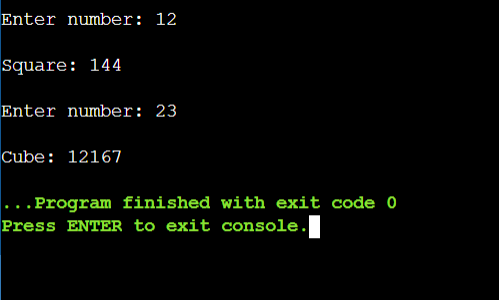
cube c;

c.c\_calc();

return 0;

}

**Output:**



**21. Aim:** Write a program to read and print employee information with department and PF information using hierarchical inheritance.

**Program:**

#include <iostream>

using namespace std;

class emp\_details

{

protected:

int emp\_no;

char phn\_no[10],emp\_name[30];

public:

void input()

{

cout<<"\nEnter Employee's name: ";

cin>>emp\_name;

cout<<"\nEnter Employee number: ";

cin>>emp\_no;

cout<<"\nEnter phone number: ";

cin>>phn\_no;

}

void ouput()

{

cout<<"\nEmployee's name: "<<emp\_name;

cout<<"\nEmployee's number: "<<emp\_no;

cout<<"\nEmployee phone number: "<<phn\_no;

}

};

class dept:public emp\_details

{

protected:

char e\_dept[20];

int w\_hours;

public:

void d\_input()

{

cout<<"\nEnter Department: ";

cin>>e\_dept;

cout<<"\nEnter working hours: ";

cin>>w\_hours;

}

void d\_output()

{

cout<<"\nDepartment: "<<e\_dept;

cout<<"\nWorking Hours: "<<w\_hours;

}

};

class details:public emp\_details

{

int basic,hra,da;

float salary;

public:

void calc\_salary()

{

cout<<"\nEnter basic, hra and da: ";

cin>>basic>>hra>>da;

salary=basic+(hra\*basic)/100+(da\*basic)/100;

cout<<"\nGross Salary: "<<salary;

}

};

int main()

{

dept d;

d.input();

d.d\_input();

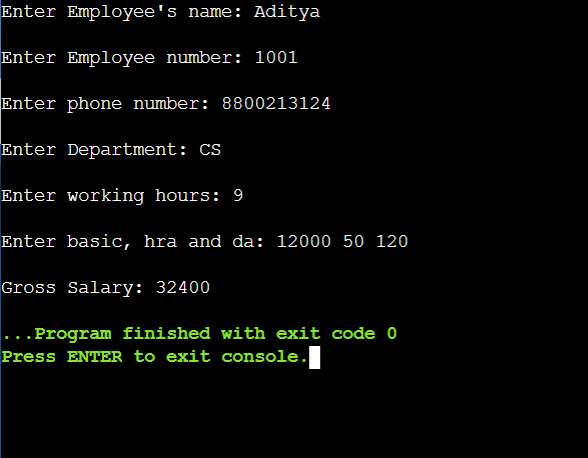
details s;

s.calc\_salary();

return 0;

}

**Output:**



**22. Aim:** Design three classes: Student, Exam and Result. The student class has data members such as roll no, name etc. Create a class Exam by inheriting the student class. The Exam class adds data members representing the marks scored in six subjects. Derive the Result from class Exam and it has its own members such as total marks. Write an interactive program to model this relationship.

**Program:**

#include <iostream>

#include <iomanip>

using namespace std;

class student

{

protected:

int roll;

char name[30];

public:

void r\_input()

{

cout<<"\nEnter Roll number: ";

cin>>roll;

cout<<"\nEnter name: ";

cin>>name;

}

void r\_output()

{

cout<<"\nRoll number: "<<roll;

cout<<"\nName: "<<name;

}

};

class test:public student

{

protected:

int marks[6];

public:

void m\_input()

{

int i;

for(i=0;i<6;i++)

{

cout<<"\nEnter marks in Subject "<<i+1<<": ";

cin>>marks[i];

}

}

void m\_output()

{

int i;

for(i=0;i<6;i++)

{

cout<<"\nMarks in Subject "<<i+1<<": ";

cout<<marks[i];

}

}

};

class result:public test

{

private:

float total,average;

public:

void res()

{

cout<<"\n----Report Card----\n";

r\_output();

m\_output();

total=0;

for(int i=0;i<6;i++)

total+=marks[i];

average=total/6;

cout<<"\nTotal: "<<total;

cout<<setw(2)<<"\nAverage: "<<average<<"%";

}

};

int main()

{

result s;

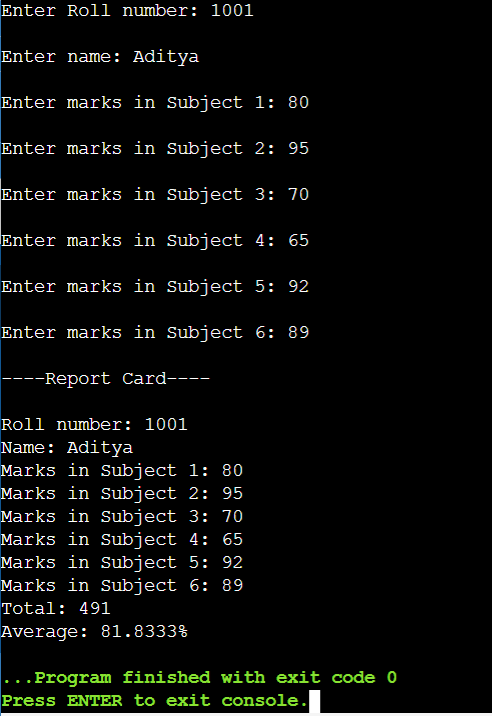
s.r\_input();

s.m\_input();

s.res();

}

**Output:**



**23. Aim:** Consider an example of book shop which sells books and video tapes. These two classes are inherited from base class called media. The media class has command data members such as title and publication. The book class has data members for storing number of pages in a book and tape class has playing time in a tape, Each class will have member functions such as read() and show(). In the base class, these members have to be defined as virtual functions. Write a program to models the class the class hierarchy for book shop and processes objects of these classes using pointers to base class.

**Program:**

#include <iostream>

#include <string>

using namespace std;

class media

{

protected:

char title[30];

float price;

public:

virtual void read() = 0;

virtual void show() = 0;

};

class book : public media

{

private:

int no\_of\_pgs;

public:

void read()

{

cout<<"\n\nEnter Book title: ";

gets(title);

cout<<"\nEnter Book price: ";

cin>>price;

cout<<"\nEnter Page count: ";

cin>>no\_of\_pgs;

}

void show()

{

cout<<"\n\n-------Book Details:-------";

cout<<"\nPublication Title: " << title;

cout<<"\nPublication Price: " << price;

cout<<"\nBook page count: " <<no\_of\_pgs;

}

};

class tape : public media

{

private:

float p\_time;

public:

void read()

{

cout<<"\n\nEnter Tape Title: ";

cin>>title;

cout<<"\nEnter Tape's price: ";

cin>>price;

cout<<"\nEnter playing time: ";

cin>>p\_time;

}

void show()

{

cout<<"\n\n-------Tape Details:-------";

cout<<"\nTape Title: " << title;

cout<<"\nTape Price: " << price;

cout<<"\nPlaying time: " << p\_time;

}

};

int main()

{

media \*b1,\*t1;

book b;

b1 = &b;

b1->read();

b1->show();

tape t;

t1 = &t;

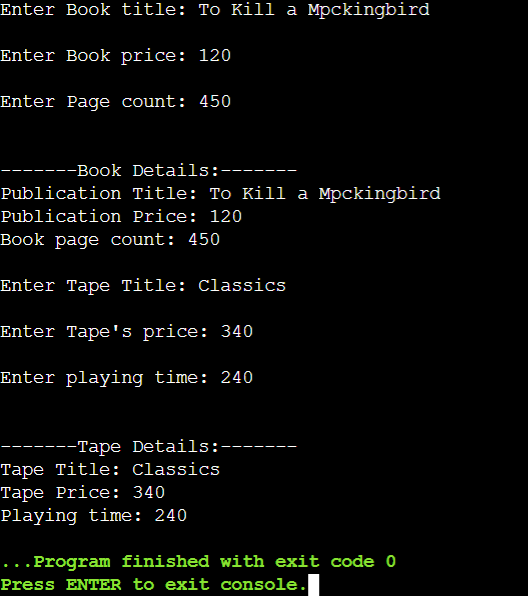
t1->read();

t1->show();

return 0;

}

**Output:**



**24. Aim:** Write a program to demonstrate operator overloading: Addition of two Complex Numbers.

**Program:**

#include<iostream>

using namespace std;

class complx

{

private:

int rl, img;

public:

complx(int r = 0, int i =0)

{

rl = r;

img = i;

}

complx operator + (complx const &p)

{

complx s;

s.rl = rl + p.rl;

s.img = img + p.img;

return s;

}

void print()

{

cout<<"Sum: "<<rl<<"+"<<img<<"i";

}

};

int main()

{

int n1,n2,ni1,ni2;

cout<<"Enter Real part of numbers you want to add: ";

cin>>n1>>n2;

cout<<"Enter Imaginary part of numbers you want to add: ";

cin>>ni1>>ni2;

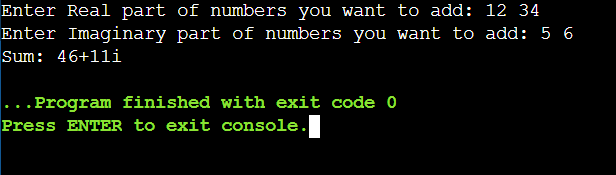
complx c1(n1, ni1), c2(n2, ni2);

complx c3 = c1 + c2;

c3.print();

}

**Output:**



**25. Aim:** Write a program to demonstrate operator overloading: Evaluating Fractions.

**Program:**

#include <iostream>

using namespace std;

class Fraction

{

int num, den;

public:

Fraction(int n, int d)

{

num = n;

den = d;

}

operator float() const

{

return float(num)/float(den);

}

};

int main()

{

int a,b;

cout<<"\nEnter numerator: ";

cin>>a;

cout<<"\nEnter denominator: ";

cin>>b;

Fraction f(a, b);

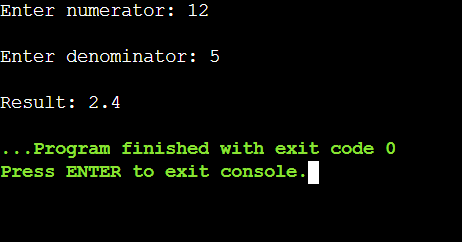
float val = f;

cout <<"\nResult: "<<val;

return 0;

}

**Output:**



**26. Aim:** Write a program to demonstrate an example of virtual functions: Displaying derived and base class virtual functions.

**Program:**

#include <iostream>

using namespace std;

class base

{

public:

virtual void print()

{

cout<<"\nBase class virtual function";

}

};

class derived:public base

{

public:

void print()

{

cout<<"\nDerived class virtual function";

}

};

int main()

{

base \*b;

base b1;derived d1;

b=&d1;

b->print();

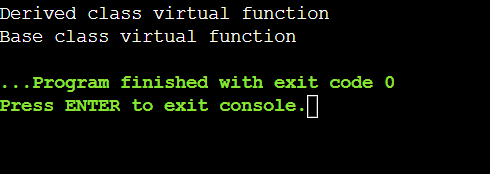
b=&b1;

b->print();

return 0;

}

**Output:**



**27. Aim:** Write a program to demonstrate an example of virtual functions: Calculating area of circle and cylinder.

**Program:**

#include <iostream>

using namespace std;

class circ

{

protected:

float r;

public:

circ(float radius)

{

r=radius;

}

virtual float area()

{

float area;

area=3.14\*r\*r;

return area;

}

};

class cyl:public circ

{

float h;

public:

cyl(float radius,float height):circ(radius)

{

h=height;

}

float area()

{

float area;

area=2\*3.14\*r\*h;

return area;

}

};

int main()

{

float r\_circ,r\_cyl,h;

cout<<"Enter Radius of circle: ";

cin>>r\_circ;

cout<<"Enter Radius and height of cyclinder: ";

cin>>r\_cyl>>h;

circ \*p;

circ c(r\_circ);

cyl co(r\_cyl,h);

p=&c;

cout<<"\nArea of Circle: "<<p->area();

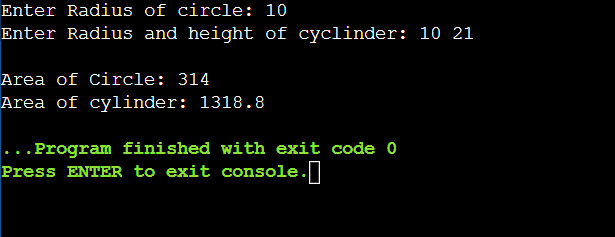
p=&co;

cout<<"\nArea of cylinder: "<<p->area();

return 0;

}

**Output:**



**28. Aim:** Write a program to demonstrate File handling.

**Program:**

#include<fstream>

#include<iostream>

using namespace std;

class student

{

    int roll,clas,admn\_no;

    char name[30];

    public:

    void enter()

    {

        cout<<"\nEnter Student's name: ";

        cin>>name;

        cout<<"\nEnter Student's admission number: ";

        cin>>admn\_no;

        cout<<"\nEnter Student's class: ";

        cin>>clas;

        cout<<"\nEnter Student's roll number: ";

        cin>>roll;

    }

    void output()

    {

        cout<<"\nStudent's name: "<<name;

        cout<<"\nStudent's admission number: "<<admn\_no;

        cout<<"\nStudent's class: "<<clas;

        cout<<"\nStudent's roll number: "<<roll<<endl;

    }

};

fstream fs;

void disp()

{

    student d;

    cout<<"\nRecords entered: ";

    fs.open("stud.dat",ios::in|ios::binary);

    while(fs.read((char \*)&d,sizeof(d)))

    {

       d.output();

    }

}

int main()

{

    int n;

    fs.open("stud.dat",ios::binary);

    student s;

    cout<<"\nEnter no of Records you want to enter:  ";

    cin>>n;

    for(int i=0;i<n;i++)

    {

        cout<<"\nRecord number: "<<i+1;

        s.enter();

        fs.write((char \*)&s,sizeof(s));

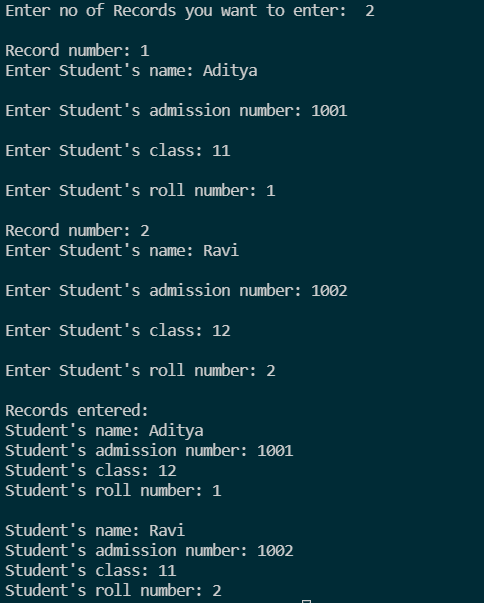
    }

    disp();

    return 0;

}

**Output:**



**29. Aim:** Write a program to demonstrate exception handling: Displaying distance.

Program:

#include <iostream>

using namespace std;

class Distance

{

private:

int feet;

float inches;

public:

class InchesEx { };

Distance()

{

feet = 0;

inches = 0.0;

}

Distance(int ft, float in)

{

if(in >= 12.0)

throw InchesEx();

feet = ft;

inches = in;

}

void getdist()

{

cout << "\nEnter feet: ";

cin >> feet;

cout << "Enter inches: ";

cin >> inches;

if(inches >= 12.0)

throw InchesEx();

}

void showdist()

{

cout << feet << " ft. " << inches << " in ";

}

};

int main()

{

char r='y';

do

{

try

{

Distance d;

d.getdist();

cout << "\nDistance:";

d.showdist();

}

catch(Distance::InchesEx)

{

cout << "\nError:value given in inch is too large.";

}

cout << endl;

cout<<"\nRepeat?(y/n)";

cin>>r;

}while(r=='Y'||r=='y');

return 0;

}

Output:

