**1. CONTROL FLOW:**

**1.1) Develop a C++ program to find the roots of a quadratic equation**

**Source code:**

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

float a,b,c,d,r1,r2;

cout<<"Enter the coefficients of a,b,c are :"<<endl;

cin>>a>>b>>c;

d=b\*b-4\*a\*c;

if(d>0)

{

r1=(-b+sqrt(d))/2\*a;

r2=(-b-sqrt(d))/2\*a;

cout<<"Roots are real and distinct"<<endl;

cout<<"r1 = "<<r1<<endl;

cout<<"r2 = "<<r2<<endl;

}

else if(d==0)

{

r1=-b/2\*a;

r2=r1;

cout<<"Roots are real and equal"<<endl;

cout<<"r1 = "<<r1<<endl;

cout<<"r2 = "<<r2<<endl;

}

else

{

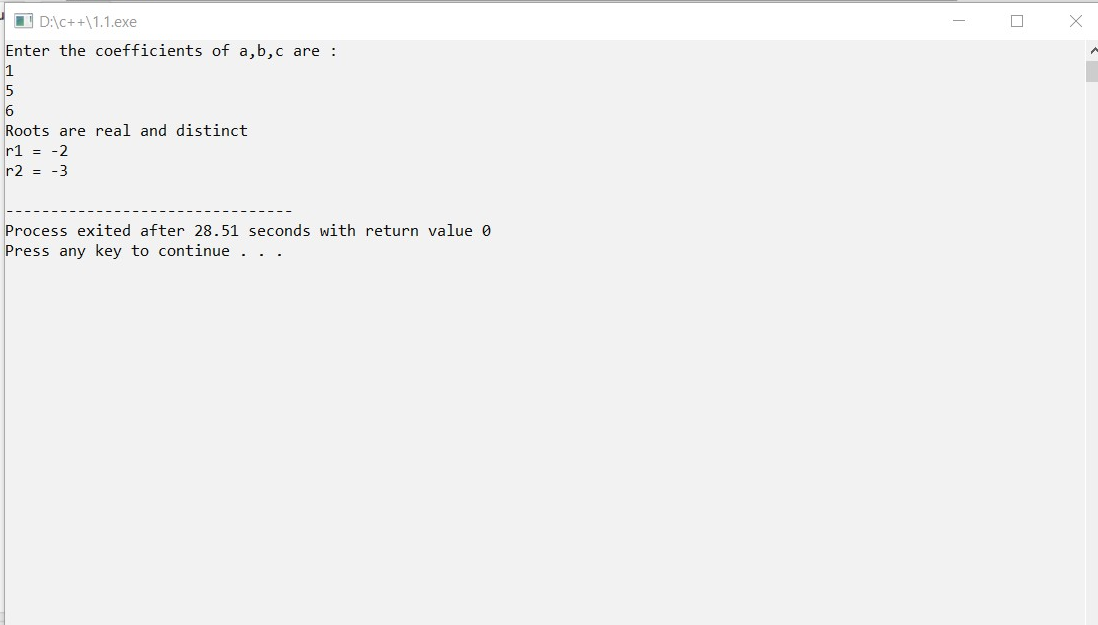
cout<<"Roots are imaginary"<<endl;

}

return 0;

}

**Output:**

****

**1.2) Develop a C++ program to find factorial of a given number using recursion**

**Source code:**

#include<iostream>

using namespace std;

int factorial(int);

int main()

{

int n;

cout<<"Enter a positive integer"<<endl;

cin>>n;

cout<<"Factorial of "<<n<<" = ";

cout<<factorial(n);

return 0;

}

int factorial(int n)

{

if(n>1)

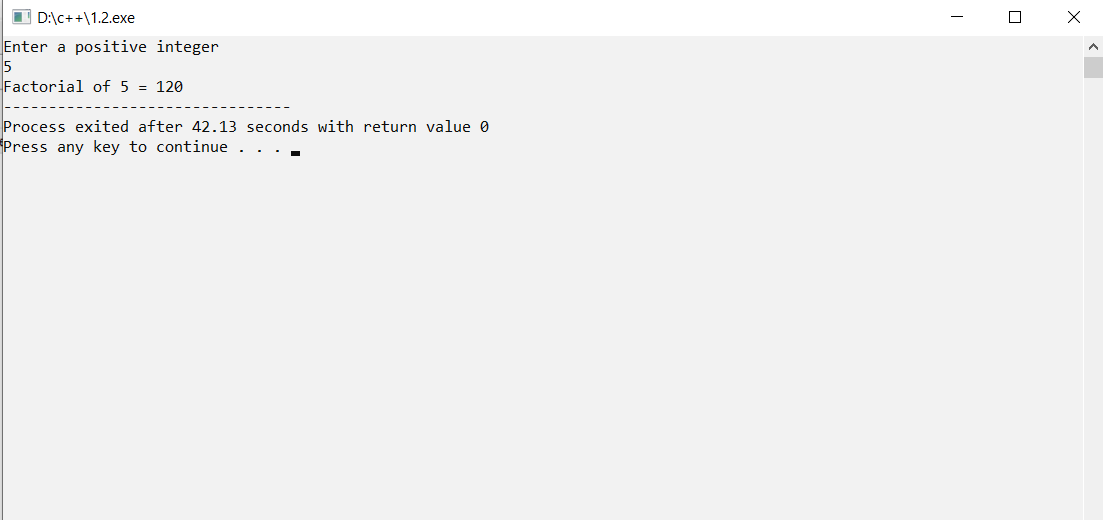
return n\*factorial(n-1);

else

return 1;

}

**Output:**

****

**2. VARAIBLE AND SCOPE:**

**2.1) Develop a C++ program to illustrate scope resolution and namespaces.**

**Source code:**

#include<iostream>

using namespace std;

namespace one

{

int a=10;

}

namespace two

{

float a=10.22;

}

int main()

{

int a=800;

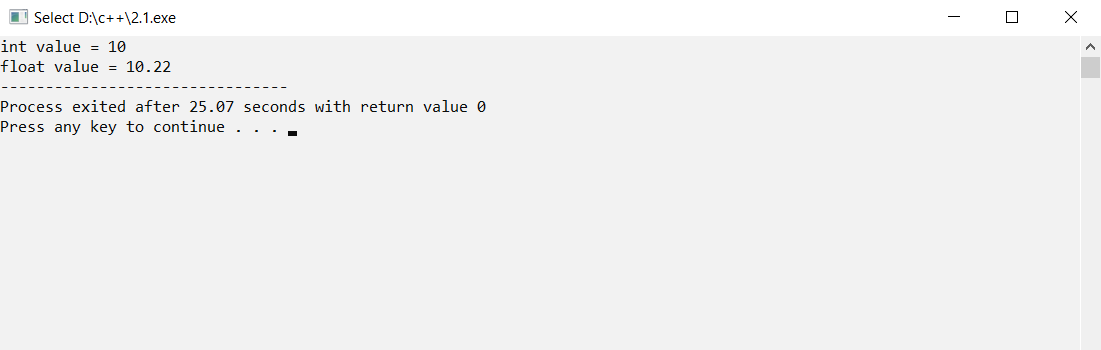
cout<<"int value = "<<one::a<<endl;

cout<<"float value = "<<two::a;

return 0;

}

**Output:**

****

**2.2) Develop a C++ program illustrating Inline Functions.**

**Source code:**

#include<iostream>

using namespace std;

int sum(int,int);

int main()

{

int a,b,c;

cout<<"enter a value ";

cin>>a;

cout<<"enter b value ";

cin>>b;

c=sum(a,b);

cout<<"sum = "<<c;

}

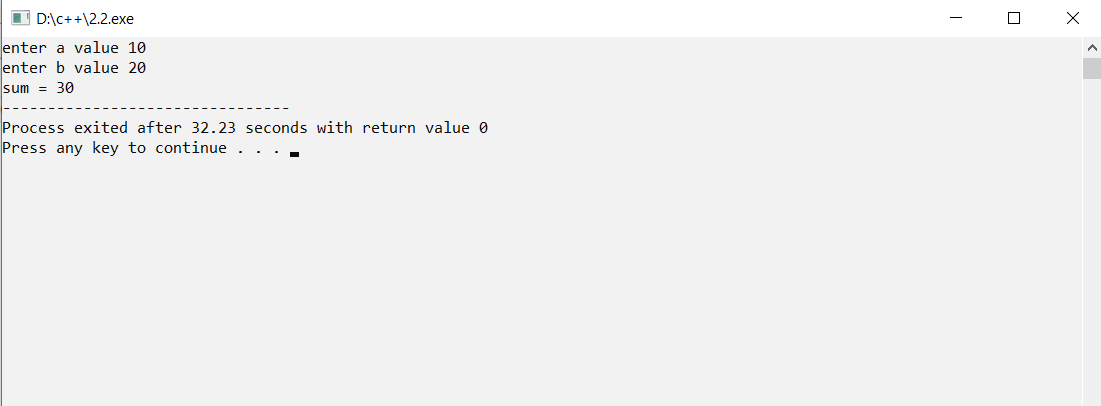
inline int sum(int x,int y)

{

return(x+y);

}

**Output:**

****

**3. CLASS AND OBJECT:**

**3.1) Develop a C++ program demonstrating a Bank Account with necessary data members and member functions.**

**Source code:**

#include<iostream>

using namespace std;

class bank

{

private:

string name;

long int mobno;

float depo;

float withd;

static float balance;

public:

void create();

void deposit();

void withdrawl();

void enquiry();

};

float bank::balance=0;

void bank::create()

{

cout<<"please enter the follwing fields"<<endl;

cout<<"name:";

cin>>name;

cout<<"mobile no";

cin>>mobno;

cout<<"your account no is \_\_\*\*\*\_\_"<<endl;

cout<<"ACCOUNT CREATED"<<endl;

float balance=0;

}

void bank::deposit()

{

cout<<"enter amount to deposit ";

cin>>depo;

balance=balance+depo;

}

void bank::withdrawl()

{

cout<<"enter amout for withdrawl ";

cin>>withd;

if(withd>balance || withd>50000)

{

cout<<"insufficient funds";

}

else

{

balance=balance-withd;

}

}

void bank::enquiry()

{

cout<<"CLOSING BALANCE IS Rs: "<<balance<<"only"<<endl;

}

int main()

{

bank b;

int op;

cout<<"WELCOME TO BANK"<<endl;

cout<<"1. create"<<"\n"<<"2. deposit"<<"\n"<<"3. withdrawl"<<"\n"<<"4. enqury"<<"\n"<<"5. exit"<<endl;

do

{

cout<<endl<<"enter your option"<<endl;

cin>>op;

switch(op)

{

case 1:b.create();

break;

case 2:b.deposit();

break;

case 3:b.withdrawl();

break;

case 4:b.enquiry();

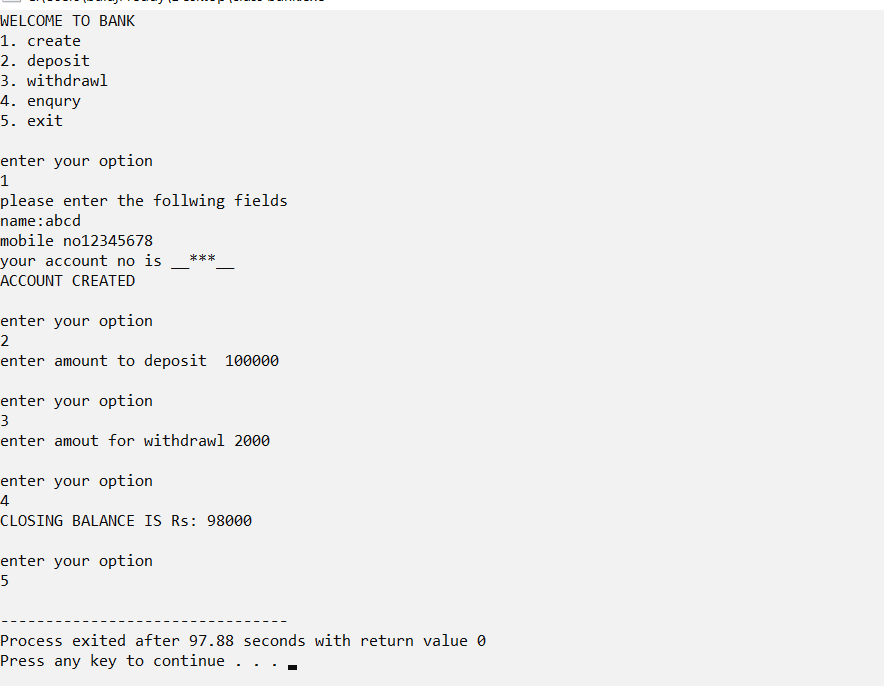
}

}while(op!=5);

return 0;

}

**Output:**

****

**3.2) Develop a C++ program for illustrating Access Specifiers :public and private.**

**Source code:**

#include<iostream>

using namespace std;

class student

{

private:

string name;

int rollno;

public:

void get()

{

cout<<"enter student name"<<endl;

cin>>name;

cout<<"enter student roll no"<<endl;

cin>>rollno;

}

friend void show(student s);

};

void show(student s)

{

cout<<"student name is "<<s.name<<endl;

cout<<"student roll no is "<<s.rollno<<endl;

}

int main()

{

student s;

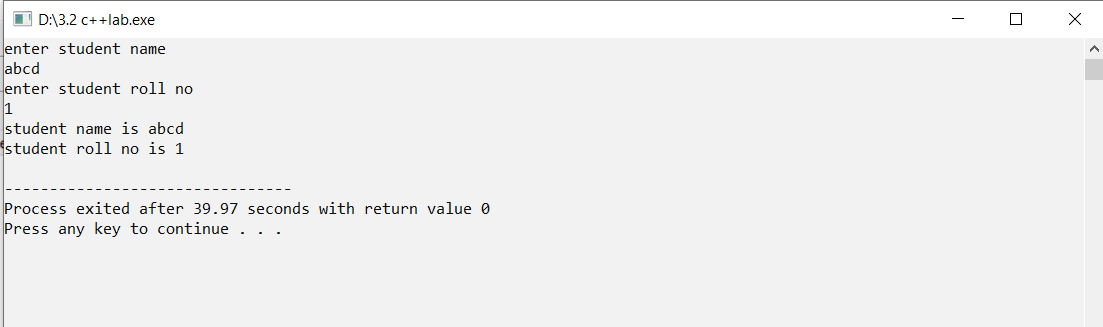
s.get();

show(s);

return 0;

}

**Output:**



**3.3) Develop a C++ program to illustrate this pointer.**

**Source code:**

#include<iostream>

using namespace std;

class rectangle

{

private:

int length,breadth;

public:

rectangle(int length,int breadth)

{

this-> length =length;

this-> breadth=breadth;

}

void area()

{

cout<<"area of rectangle = "<<length\*breadth<<endl;

}

};

int main()

{

int l,b;

cout<<"enter lenght of rectangle ";

cin>>l;

cout<<"enter breadth of rectangle ";

cin>>b;

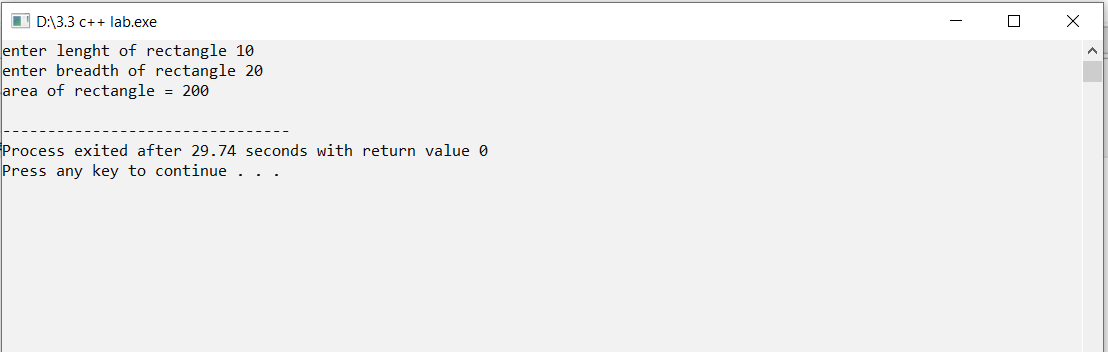
rectangle r(l,b);

r.area();

return 0;

}

**Output:**



**4. FUNCTIONS:**

**4.1) Develop a C++ program illustrate function overloading**

**Source code:**

#include<iostream>

using namespace std;

class sample

{

public:

void display(int x)

{

cout<<"x = "<<x<<endl;

}

void display(float y)

{

cout<<"y = "<<y<<endl;

}

void display(char c)

{

cout<<"c = "<<c<<endl;

}

void display(string str)

{

cout<<"str = "<<str<<endl;

}

void display(int a,int b)

{

cout<<"a = "<<a<<"b = "<<b;

}

};

int main()

{

sample s;

s.display(10);

s.display(3.56f);

s.display('a');

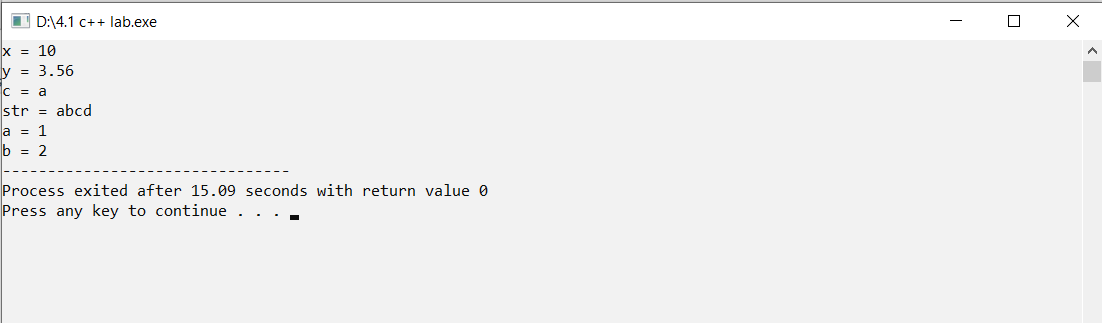
s.display("abcd");

s.display(1,2);

return 0;

**}**

**Output:**

****

**4.2) Develop a C++ program to illustrate the use of default arguments.**

**Source code:**

#include<iostream>

using namespace std;

void intrest(float p=10,float t=2,float r=3)

{

float si;

si=(p\*t\*r)/100;

cout<<"simple intrest = "<<si<<endl;

}

int main()

{

float p=100,t=2,r=0.5;

intrest();

intrest(p,t,r);

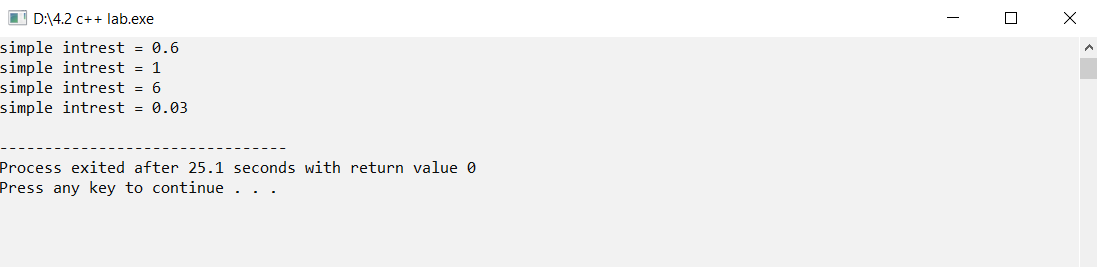
intrest(p,t);

intrest(r);

return 0;

}

**Output:**

****

**4.3) Develop a C++ program illustrating friend function.**

**Source code:**

#include<iostream>

using namespace std;

class sample

{

private:

int a, b;

public:

void get()

{

cout<<"enter a ,b";

cin>>a>>b;

}

friend void sum(sample s); //declaration of friend function

};

void sum(sample s)

{

cout<<"sum is "<<s.a+s.b; //friend function definition

}

int main()

{

sample c;

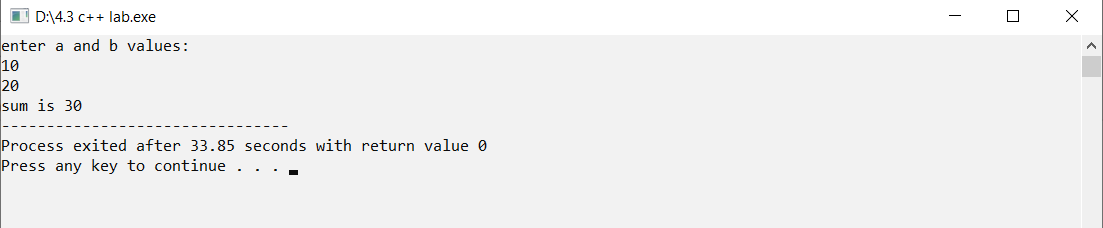
c.get();

sum(c);

return 0;

}

**Output:**

****

**5. CONSTRUCTOR AND DESTRUCTOR**:

**5.1) Develop a C++ Program to illustrate the use of Constructors and Destructors.**

**Source code:**

#include<iostream>

using namespace std;

class construct

{

public:

construct()

{

cout<<"constuctor invoked"<<endl;

}

~construct()

{

cout<<"destuctor invoked"<<endl;

}

};

int main()

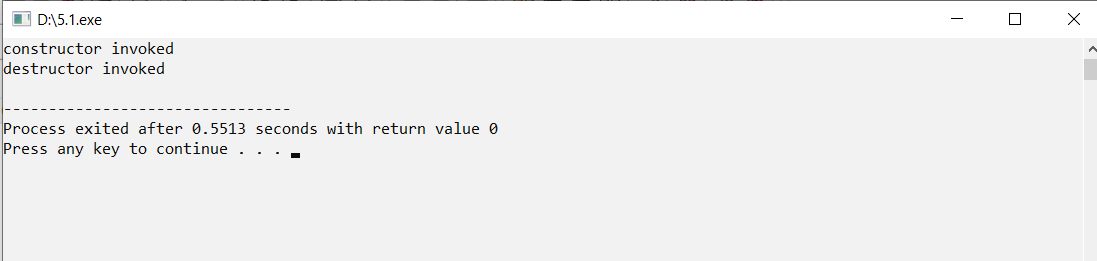
{

construct c;

return 0;

}

**Output:**



**5.2) Develop a C++ program illustrating Constructor overloading.**

**Source code:**

#include<iostream>

using namespace std;

class rectangle

{

private:

float length,breadth;

public:

rectangle()

{

length=10;

breadth=20;

}

rectangle(int l,int b)

{

length=l;

breadth=b;

}

rectangle(float l,float b)

{

length=l;

breadth=b;

}

void area()

{

cout<<"area of rectangle = "<<length\*breadth<<endl;

}

};

int main()

{

rectangle r1;

rectangle r2(2,3);

rectangle r3(0.5f,0.5f);

r1.area();

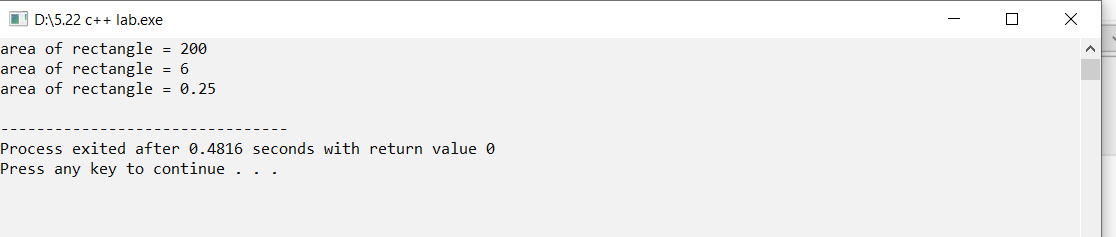
r2.area();

r3.area();

return 0;

}

**Output:**



**5.3) Develop a C++ program illustrating Copy Constructor.**

**Source code:**

#include<iostream>

using namespace std;

class sample

{

private:

int a,b;

public:

sample(int a,int b)

{

this->a=a;

this->b=b;

}

sample(sample &old)

{

a=old.a;

b=old.b;

}

void show()

{

cout<<"a= "<<a<<endl;

cout<<"b= "<<b<<endl;

}

};

int main()

{

sample s(10,20);

s.show();

cout<<endl;

sample s2(s);

s2.show();

return 0;

}

**Output:**

