**-------------------------------------------------------------**

NAME  **:** Atharv Santosh Ghule

BRANCH **:** FYBsc.IT

DIV  **:** A

ROLL NO  **:** 1904

SUBJECT  **:** Cpp Programming.

**-------------------------------------------------------------**

Cpp PROGRAMS PRACTICAL FILE

* Program: 1

**Source Code:**

*//To print Hello World!  1904*

#include <iostream>

using namespace std;

int main()

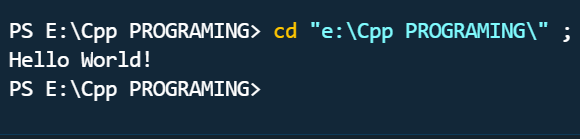
{

    cout<<"Hello World!";

    return 0;

}

**OUTPUT:**



* Program: 2

**Source Code:**

*//Program for factorial or reverse number  1904*

#include<iostream>

using namespace std;

int main()

{

    int n, reverse=0, rem;

    cout<<"Enter a number:";

    cin>>n;

    while(n!=0)

    {

        rem=n%10;

        reverse=reverse\*10+rem;

        n/=10;

    }

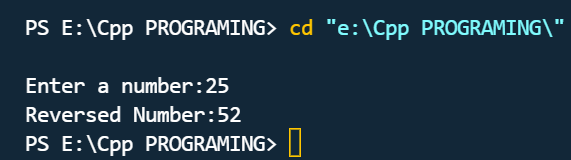
    cout<<"Reversed Number:"

    <<reverse<<endl;

    return 0;

}

**OUTPUT:**



* Program: 3 (Classes & Methods)

**Source Code:**

*//Program for employee using classes & objects  1904*

#include<iostream>

using namespace std;

class *employee*

{

    private:

    int id,salary;

    char name[20];

    public:

    void getinfo()

    {

        cout<<"Enter your name,id & salary:";

        cin>> name>> id >>salary;

    }

    void display()

    {

        cout<<"Your Name is:"<<name<<endl

            <<"Your salary:"<<salary<<endl;

        cout <<"Your id = "<<id;

    }

};

int main ()

{

*employee* e;

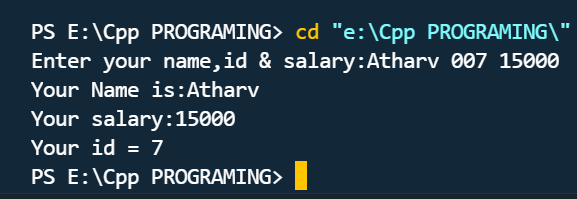
    e.getinfo();

    e.display();

return 0;

}

**OUTPUT:**

****

* Program: 4 (Classes & Methods)

**Source Code:**

*//Program for StudentDetails using classes & object  1904*

#include<iostream>

using namespace std;

class *student*

{

    private:

    char name[20];

    int rollno;

    float Perc;

    private:

    void getdata()

    {

      cout<<"Enter your Name, Roll no, Percentage";

        cin>> name>> rollno >>Perc;

    }

    public:

    void showdata()

    {

        getdata();

        cout<<"Your Name is:"<<name;

        cout<<endl<<"Your Rollno is:"<<rollno;

        cout<<endl<<"Your Percentage is:"<<Perc;

    }

};

int main()

{

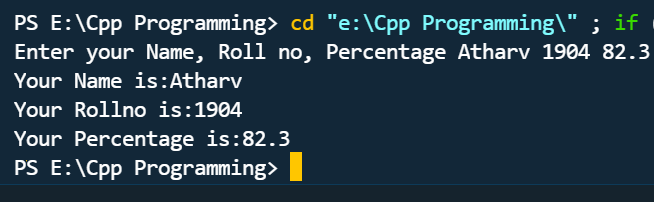
*student* s;

    s.showdata();

    return 0;

}

**OUTPUT:**

****

* Program: 5 (Classes & Methods)

**Source Code:**

*//PROGRAM FOR CLASS DEMO 1904*

#include<iostream>

#include<math.h>

using namespace std;

class Demo

{

int num, len=0;

public:

Demo()

{

readNo();

}

void factorial()

{

int a = num;

int i = 1;

while(a>0)

{

i = i \* a;

a--;

len++;

}

cout<<"Factorial of "<<num<<" is "<<i;

}

void reverse()

{

int a = num;

int rev = 0;

while(a>0)

{

rev = rev\*10 + a%10;

a /= 10;

}

cout<<endl<<"nReverse of "<<num<<" is "<<rev;

}

void isPalindrome()

{

int a = num;

int rev = 0;

while(a>0)

{

rev =rev\*10 + a%10;

a /= 10;

}

if(rev == num)

{

cout<<endl<<"nYes!!! It is a Paliindrome number";

}

else

{

cout<<endl<<"nNo!!! It is NOT a Paliindrome number";

}

}

void isArmstrong()

{

int a = num;

int n = num;

int rem = 0, rsl=0, i = 1;

while(a>0)

{

a/=10;

len++;

}

while(n>0)

{

rem = n%10;

rsl += pow(rem,len);

n /= 10;

}

if(rsl == num)

{

cout<<endl<<"nYes!!! It is an Armstrong number";

}

else

{

cout<<endl<<"nNo!!! It is NOT an Armstrong number";

}

}

private:

void readNo()

{

cout<<"Enter any number : ";

cin>>num;

}

};

int main()

{

Demo d = Demo();

d.factorial();

d.reverse();

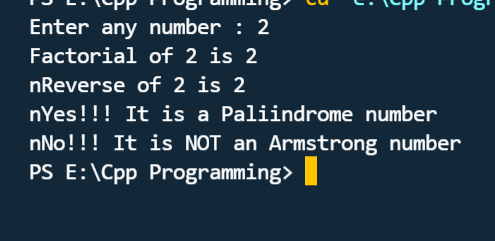
d.isPalindrome();

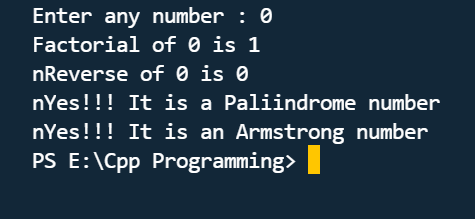
d.isArmstrong();

return 0;

}

**OUTPUT:**





* Program: 6 (using friend functions)

**Source Code:**

*// PROGRAM TO ADD TWO COMPLEX NUMBERS   1904*

#include<iostream>

using namespace std;

class *Complex*{

public:

    int real;

    int imag;

*// Function to set the values of real and imaginary part of each complex number*

     void setvalue()

    {

        cin>>real;

        cin>>imag;

    }

*//Function to display the sum of two complex numbers*

    void display()

    {

        cout<<real<<"+"<<imag<<"i"<<endl;

    }

*//Function to add two complex numbers*

    void sum(*Complex* c1, *Complex* c2)

    {

        real=c1.real+c2.real;

        imag=c1.imag+c2.imag;

    }

    };

int main()

    {

*Complex* c1,c2,c3;

        cout<<"Enter real and imaginary part of first complex number"<<endl;

        c1.setvalue();

        cout<<"Enter real and imaginary part of second complex number"<<endl;

        c2.setvalue();

        cout<<"Sum of two complex numbers is"<<endl;

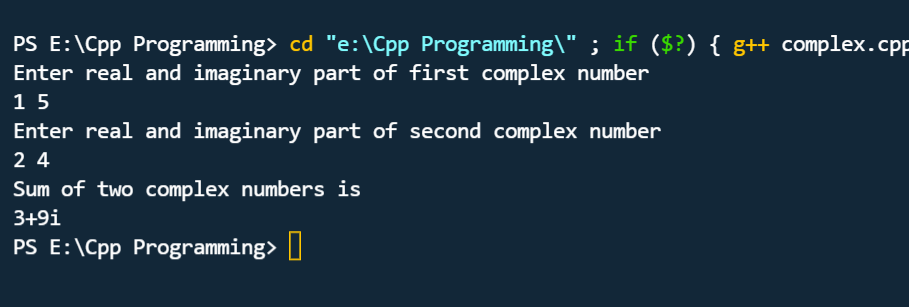
        c3.sum(c1,c2);

        c3.display();

    return 0;

    }

**OUTPUT:**



* Program: 7(using friend functions)

**Source Code:**

*//Program for different distances and display its  sum, using two classes. 1904*

#include<iostream>

using namespace std;

class *dist2*;

class *dist1*

{

    public:

        float mtr, cm;

    public:

        void accept()

        {

                cout<<"\n Enter Data in Meter & Centimeter  :  ";

                cin>>mtr>>cm;

        }

*friend* void diff(*dist1* d1, *dist2* d2);

*friend* void sum(*dist1* d1, *dist2* d2);

};

class *dist2*

{

        float feet, inch;

    public:

        void accept()

        {

                cout<<"\n Enter Data in Feet & Inch         :  ";

                cin>>feet>>inch;

        }

*friend* void difference(*dist1* d1, *dist2* d2);

*friend* void sum(*dist1* d1, *dist2* d2);

};

void difference(*dist1* d1,*dist2* d2)

{

        int n1, n2, n3, ans, m, c, f, in;

        n1=d2.inch\*2.54;

        n2=d2.feet\*0.30;

        n3=d1.mtr\*100;

        ans=((d1.cm + n3) - (n1 + n2));

        m=ans/100;

        c=ans%100;

        cout<<"\n --------------------------------------------------------------------";

        cout<<"\n Difference in Meters & Centimeters  =  "<<m<<" mtrs & "<<c<<" cms";

        f=m/0.30;

        in=c/2.54;

        cout<<"\n Difference in Feets & Inches        =  "<<f<<" feets & "<<in<<" inches";

}

void sum(*dist1* d1, *dist2* d2)

{

        int n1, n2, n3, ans, m, c, f, in;

        n1=d2.inch\*2.54;

        n2=d2.feet\*0.30;

        n3=d1.mtr\*100;

        ans=((d1.cm + n3) + (n1+n2));

        m=ans/100;

        c=ans%100;

        cout<<"\n ------------------------------------------------------------------";

        cout<<"\n Sum in Meters & Centimeters         =  "<<m<<" mtrs & "<<c<<" cms";

        f=m/0.30;

        in=c/2.54;

        cout<<"\n Sum in Feets & Inches               =  "<<f<<" feets & "<<in<<" inches";

}

int main()

{

*dist1* d1;

*dist2* d2;

        d1.accept();

        d2.accept();

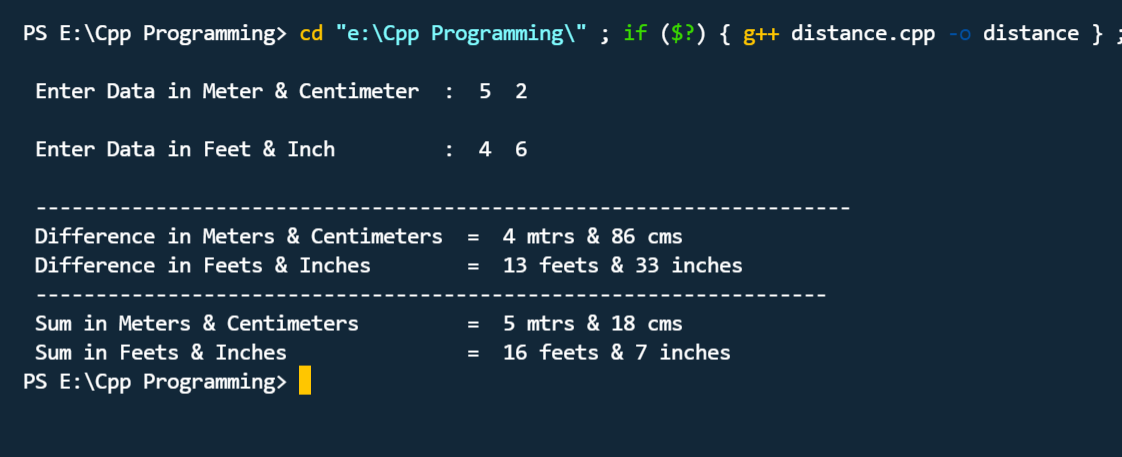
        difference(d1,d2);

        sum(d1,d2);

        return 0;

}

**OUTPUT:**



* Program: 8 (using friend functions)

**Source Code:**

*//PROGRAM FOR ADDING THE TWO MATRIX FROM TWO DIFFERENT CLASSES  1904*

#include <iostream>

using namespace std;

int main()

{

int r, c, a[100][100], b[100][100], sum[100][100], i, j;

cout << "Enter number of rows (between 1 and 100): ";

cin >> r;

cout << "Enter number of columns (between 1 and 100): ";

cin >> c;

cout << endl << "Enter elements of 1st matrix: " << endl;

// Storing elements of first matrix entered by user.

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

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

{

cout << "Enter element a" << i + 1 << j + 1 << " : ";

cin >> a[i][j];

}

// Storing elements of second matrix entered by user.

cout << endl << "Enter elements of 2nd matrix: " << endl;

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

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

{

cout << "Enter element b" << i + 1 << j + 1 << " : ";

cin >> b[i][j];

}

// Adding Two matrices

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

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

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

// Displaying the resultant sum matrix.

cout << endl << "Sum of two matrix is: " << endl;

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

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

{

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

if(j == c - 1)

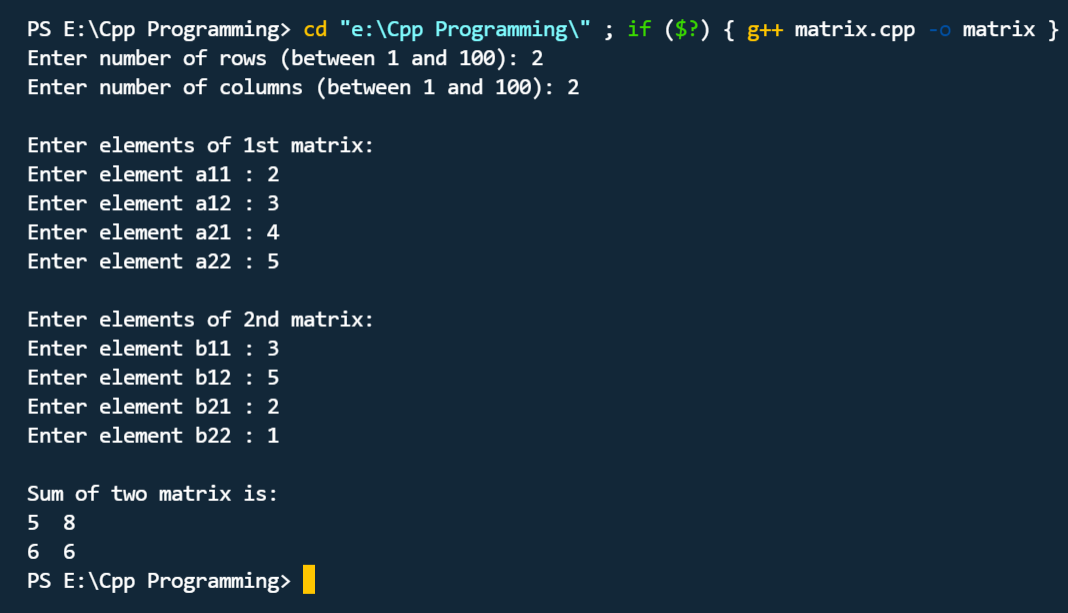
cout << endl;

}

return 0;

}

**OUTPUT:**

****

* Program: 8 (Constructors and method overloading)

**Source Code:**

*//PROGRAM FOR UNARY 1904*

#include<iostream>

#include<conio.h>

using namespace std;

class *Negate*

{

    int x,y;

    public:

    void read()

    {

        cout<<"Enter two numbers";

        cin>>x>>y;

    }

    void operator -()

    {

        x=-x;

        y=-y;

    }

    void display()

    {

        cout<<"x="<<x<<endl<<"y="<<y;

    }

};

int main()

{

*Negate* n;

    n.read();

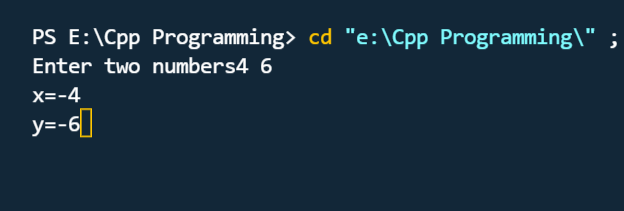
    -n;

    n.display();

    getch();

}

**OUTPUT :**

****

* Program: 9 (Constructors and method overloading)

**Source Code:**

*//PROGRAM FOR CLASS GEOMETRY  1904*

#include<iostream>

using namespace std;

class *geometry*

{

    public:

        int area(int x)

        {

            return(x\*x);

        }

        int area(int x,int y)

        {

            return(x\*y);

        }

        int volume(int x)

        {

            return(x\*x\*x);

        }

        int volume(int x, int y, int z)

        {

            return(x\*y\*z);

        }

};

int main()

{

*geometry* g;

    cout<<"Area of Square is "<<g.area(20)<<endl;

    cout<<"Area of rectangle is "<<g.area(30,12)<<endl;

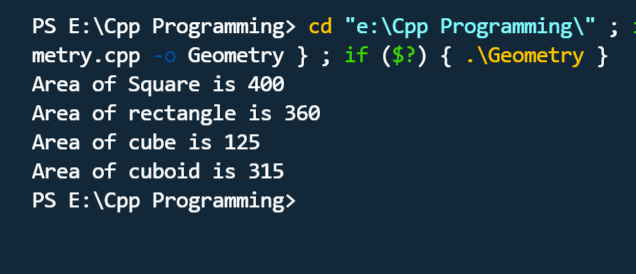
    cout<<"Area of cube is "<<g.volume(5)<<endl;

    cout<<"Area of cuboid is "<<g.volume(5,7,9);

    return 0;

}

**OUTPUT :**



* Program: 10 (Operator Overloading)

**Source Code:**

*//PROGRAM FOR CALCULATING TIME  1904*

#include<iostream>

using namespace std;

class *time2*;

class *time1*

{

    int hr,min;

    public:

    void gettime(int a,int b)

    {

        hr=a;

        min=b;

    }

*friend* void operator+(*time1*,*time2*);

};

class *time2*

{

    int h,m;

    public:

    void getdata(int p,int q)

    {

        h=p;

        m=q;

    }

*friend* void operator+(*time1*,*time2*);

};

void operator+(*time1* obj1, *time2* obj2)

{

    int minute,hour;

    hour=obj1.hr+obj2.h+(obj1.min+obj2.m)/60;

    minute=(obj1.min+obj2.m)%60;

    cout<<"\n Total time from both classes:\n";

    cout<<hour<<"hrs"<<minute<<"mins";

}

int main()

{

*time1* t1;

*time2* t2;

    cout<<"\n Program to add timing of two classes \n";

    cout<<"\n Timing from clock1:\n";

*//cout<<"hour="<<hr<<"minute="<<min;*

    cout<<"\n Timing from clock2:\n";

*//cout<<"hour="<<hr<<"minutes=\n"<<m;*

    t1.gettime(100,20);

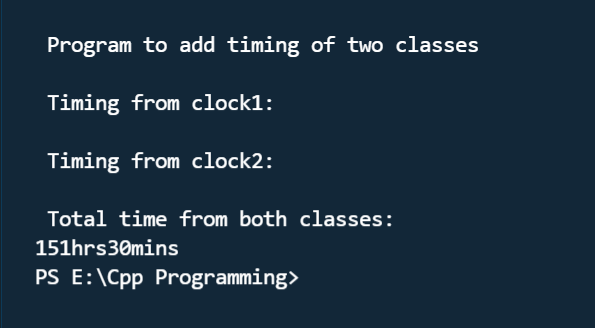
    t2.getdata(50,70);

    t1+t2;

    return 0;

}

**OUTPUT :**

****

* Program: 11 (Operator Overloading)

**Source Code:**

#include<iostream>

#include<conio.h>

using namespace std;

class *Negate*

{

    int x,y;

    public:

    void read()

    {

        cout<<"Enter two numbers";

        cin>>x>>y;

    }

    void operator -()

    {

        x=-x;

        y=-y;

    }

    void display()

    {

        cout<<"x="<<x<<endl<<"y="<<y;

    }

};

int main()

{

*Negate* n;

    n.read();

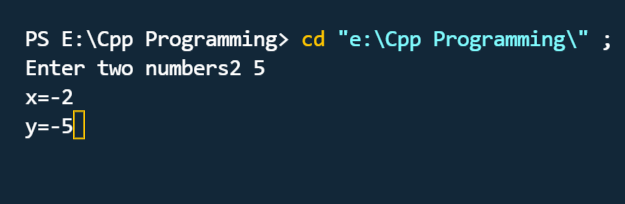
    -n;

    n.display();

    getch();

}

**OUTPUT :**



* Program: 12 (Operator Overloading)

**Source Code:**

*//PROGRAM FOR CLASS STRINGS  1904*

#include<iostream>

#include<string.h>

using namespace std;

class *Addstring*

{

    public:

char s1[25],s2[25];

Addstring(char str1[],char str2[])

{

    strcpy(this->s1, str1);

    strcpy(this->s2, str2);

}

void operator+()

{

    cout<<"\n Concatenation:"<<strcat(s1,s2);

}

};

int main()

{

    char str1[]="Py";

    char str2[]="thon";

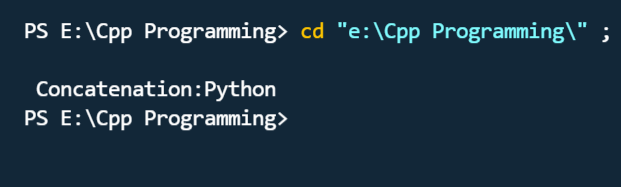
*Addstring* a1(str1, str2);

    +a1;

    return 0;

}

**OUTPUT :**



* Program: 13 (Inheritance)

**Source Code:**

*//PROGRAM FOR SINGLE INHERTANCE  1904*

#include<iostream>

using namespace std;

class *base*

{

    int x;

    public :

    void getdata()

    {

        cout<<"Enter the value of x=";

        cin>>x;

    }

};

class *derive* : public *base*

{

private :

int y;

public :

void readdata()

{

    cout<<"Enter the value of y=";

    cin>>y;

}

void product()

{

    cout<<"Product:<<x";y;

}

};

int main()

{

*derive* a;

    a.getdata();

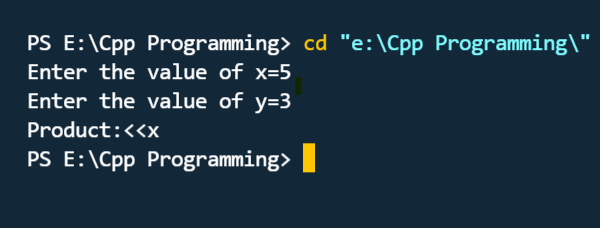
    a.readdata();

    a.product();

    return 0;

}

**OUTPUT :**



* Program: 13 (Inheritance)

**Source Code:**

*//PROGRAM FOR MULTIPLE INHERITANCE  1904*

#include<iostream>

using namespace std;

class *person*

{

protected:

  char name[15];

  protected:

  void accept()

  {

    cout<<"Enter your name:";

    cin>>name;

  }

};

class *student*

{

  protected:

 int rollno=1904;

};

class *Example*:protected *person*,protected *student*

{

    public:

    void display()

    {

      accept();

     cout<<"\nYour name and rollno are:"<<name<<"\n"<<rollno;

    }

};

int main()

{

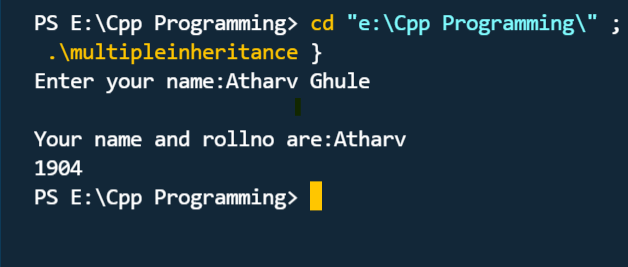
*Example* e;

e.display();

return 0;

}

**OUTPUT :**

****

* Program: 14 (Inheritance)

**Source Code:**

*//PROGRAM FOR hierarchical inheritance. 1904*

#include <iostream>

using namespace std;

class *A*

{

    public:

    int x, y;

    void getdata()

    {

        cout<<"\nEnter value of x and y:";

        cin>>x>> y;

    }

};

class *B* : public *A*

{

    public:

    void Add()

    {

        cout << "\nAddition:"<<x+y;

    }

};

class *C* : public *A*

{

    public:

    void sub()

    {

        cout << "\nSubtraction: "<<x-y;

    }

};

int main()

{

*B* obj1;

*C* obj2;

    obj1.getdata();

    obj1.Add();

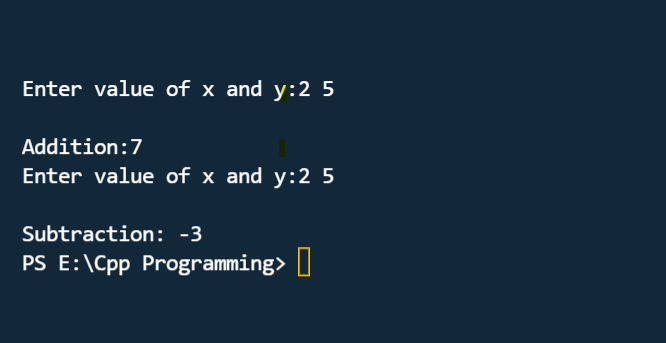
    obj2.getdata();

    obj2.sub();

    return 0;

}

**OUTPUT :**



* Program: 15 (Virtual functions and abstract classes)

**Source Code :**

*//PROGRAM FOR OVERRIDING  1904*

#include <iostream>

using namespace std;

class *Base*

{

   public:

   void print()

   {

    cout<<"Atharv";

    }

};

class *Derived*:public *Base*

{

   public:

   void print()

   {

    cout<<"Ghule";

    }

};

int main()

{

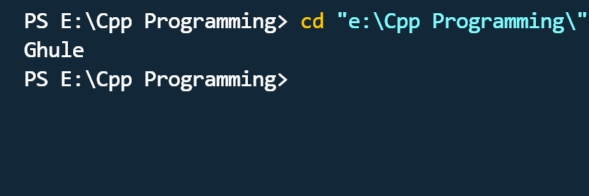
*Derived* derived1;

    derived1.print();

    return 0;

}

**OUTPUT :**



* Program: 16 (Virtual functions and abstract classes)

**Source Code :**

*//PROGRAM FOR VIRTUAL FUNCTION  1904*

#include<iostream>

using namespace std;

class *base*

{

    public:

*virtual* void display()

    {

        cout<<"\nhello";

    }

};

class *derived*:public *base*

{

    public:

    void display()

    {

        cout<<"\nATHARV";

    }

};

int main()

{

*base*\*b,a;

*derived* d;

    b=&d;

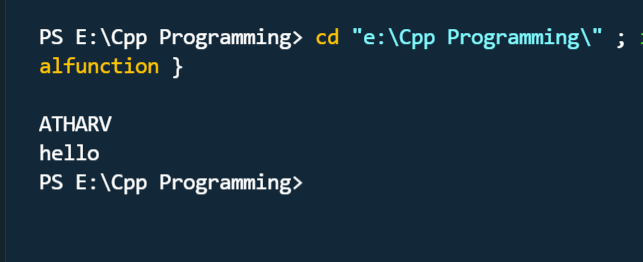
    b->display();

    a.display();

    return 0;

}

**OUTPUT :**

****

* Program: 17 (Virtual functions and abstract classes)

**Source Code :**

*//PROGRAM FOR IMPLEMENTATION OF ABSTRACT CLASS  1904*

#include<iostream>

using namespace std;

class *base*

{

    public:

*virtual* void display()=0;

};

class *derived*

{

    public:

    void display()

    {

        cout<<"\nAtharv";

    }

};

class *derive*: public *derived*

{

    public:

    void display()

    {

        cout<<"\nGhule";

    }

};

int main()

{

*derived* d;

    d.display();

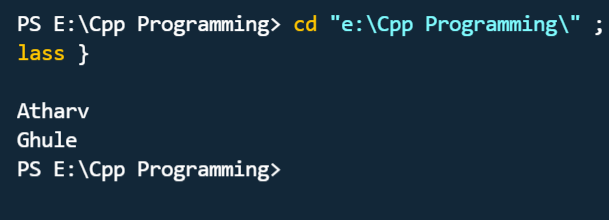
*derive* c;

    c.display();

    return 0;

}

**OUTPUT :**

****

* Program: 18 (String handling)

**Source Code:**

*//PROGRAM FOR STRING LENGTH , CONCENTATION  1904*

#include <iostream>

#include<string.h>

using namespace std;

class *sci*

{

char x[20]="science";

int len;

public:

void get()

{

cout<<"before try"<<endl;

len=strlen(x);

cout<<"length of string is : "<<len<<endl;

try

{

cout<<"inside try "<<endl;

if(len>5)

{

throw x;

cout<<"after throw (never executed)"<<endl; }

}

catch(char x)

{

cout<<"exception caught"<<endl;

cout<<"length of string is : "<<len<<endl;

cout<<"inside catch"<<endl;

}

}

};

int main()

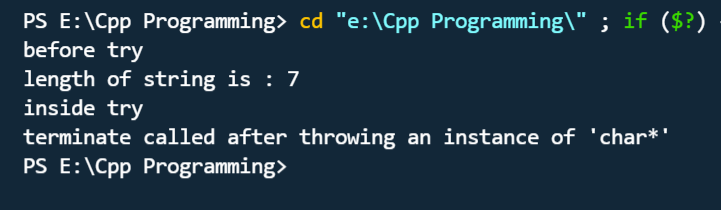
{

*sci* a;

a.get();

}

**OUTPUT:**

****

* Program: 19 (String handling)

**Source Code:**

*//Program for String operations for string reverse  1904*

#include <iostream>

using namespace std;

int main()

{

*string* str1="",

str2="";

 cout<<"Enter String 1:";

 cin>>str1;

 cout<<"\nEnter String 2:";

 cin>>str2;

 str1.append(str2);

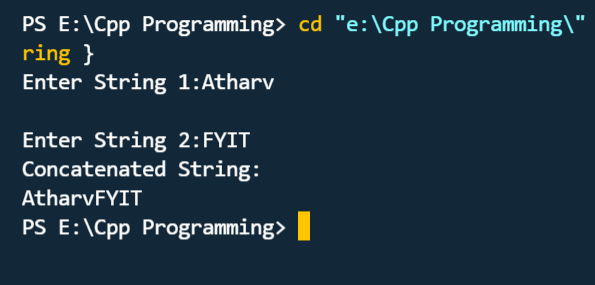
 cout<<"Concatenated String:"<<endl;

 cout<<str1;

 return 0;

}

**OUTPUT:**



* Program: 20(String handling)

**Source Code:**

*//PROGRAM FOR REVERSE STRING  1904*

#include<iostream>

#include<string.h>

#include<stdio.h>

using namespace std;

int main()

{

 char str[100], temp;

 int i=0, j;

 cout<<"\n Enter String : ";

 gets(str);

 i=0;

 j=strlen(str)-1;

 while(i<j)

 {

 temp=str[i];

 str[i]=str[j];

 str[j]=temp;

 i++;

 j--;

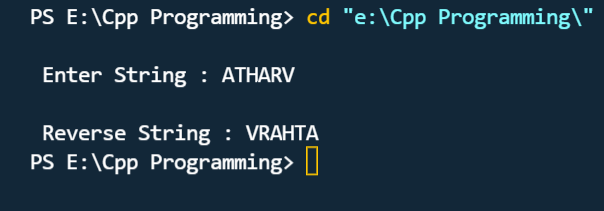
 }

 cout<<"\n Reverse String : "<<str;

 return 0;

}

**OUTPUT:**



* Program: 21(String handling)

**Source Code:**

*//PROGRAM FOR STRING COMPARISON  1904*

#include<iostream>

using namespace std;

int main()

{

 string str1="Welcome to javatpoint";

 string str2="Welcome to javatpoint";

 int i=str1.compare(str2);

 if(i==0)

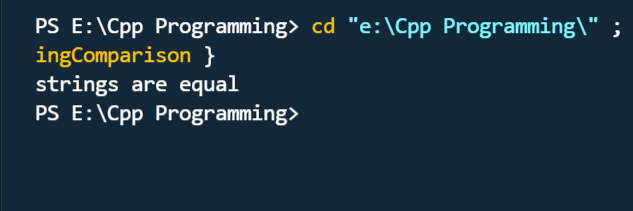
 cout<<"strings are equal";

else

cout<<"strings are not equal";

}

**OUTPUT:**



* Program: 22 (Exceptional Handling)

**Source Code:**

*//PROGRAM FOR EXCEPTIONAL HANDLING 1904*

#include<iostream>

using namespace std;

class *Percentage*

{

    int x =0;

    public :

    void get()

    {

        cout<<"Enter your Percentage";

        cin>>x;

    }

    void show()

    {

        try{

            if (x < 0 || x>100)

            {

                throw x;

                cout<<"\n After throw statement that is never executed\n";

            }

            }

            catch (int x)

            {

                cout<< "Exception Caught,The percentage entered is NOT VALID !\n";

            }

            cout<<"Your Percentage is :"<<x;

    }

};

int main()

{

*Percentage* p;

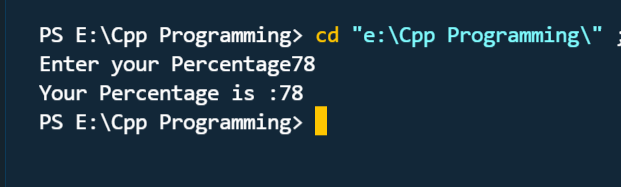
    p.get();

    p.show();

    return 0;

}

**OUTPUT:**



* Program: 23 (Exceptional Handling)

**Source Code:**

*//Program for exception  handling for strings  1904*

#include<iostream>

using namespace std;

void Evenodd(int n)

{

    try

    {

        if(n%2==0)

        throw 1;

        else

        throw 1.0;

    }catch(...)*//catch all exceptions*

    {

        cout<<"EXCEPTION ENCOUNTERED !!";

    }

}

int main()

{

    int a;

    cout<<"Enter a Number";

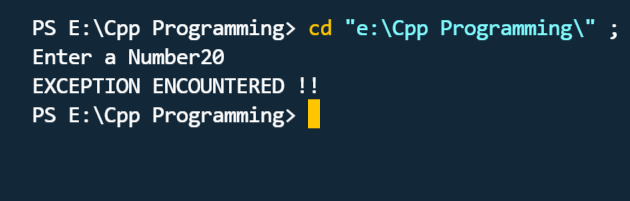
    cin>>a;

    Evenodd(a);

    return 0;

}

**OUTPUT:**



* Program: 24(Exceptional Handling)

**Source Code:**

*//PROGRAM FOR TRY , CATCH & THROW  1904*

#include<iostream>

using namespace std;

int main()

{

   int x = -1;

cout<< "Before try \n";

   try {

cout<< "Inside try \n";

      if (x < 0)

      {

         throw x;

cout<< "After throw (Never executed) \n";

      }

   }

   catch (int x ) {

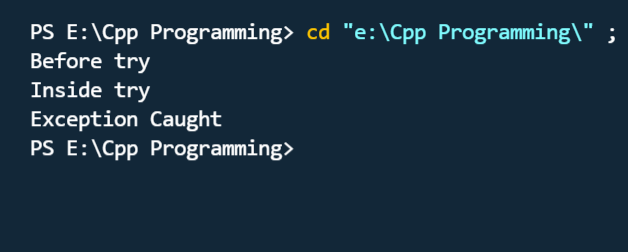
cout<< "Exception Caught \n";

   }

return 0;

}

**OUTPUT:**

****

* Program: 25 (File Handling)

**Source Code:**

#include<iostream>

#include<string.h>

using namespace std;

int main()

{

 int noc=0,now=0,nol=0;

*FILE* \*fr;

 char fname[20],ch;

 cout<<"\n Enter Source File Name : ";

 gets(fname);

 fr=fopen(fname,"r");

 if(fr==NULL)

 {

 cout<<"\n Invalid File Name. \n No such File or Directory ";

 exit(0);

 }

 ch=fgetc(fr);

 while(ch!=EOF)

 {

 if(ch!=' ' && ch!='\n')

 noc++;

 if(ch==' ')

 now++;

 if(ch=='\n')

 {

 nol++;

 now++;

 }

 ch=fgetc(fr);

}

 fclose(fr);

 cout<<" -------------------------------------";

 cout<<"\n Total No. of Characters : "<<noc;

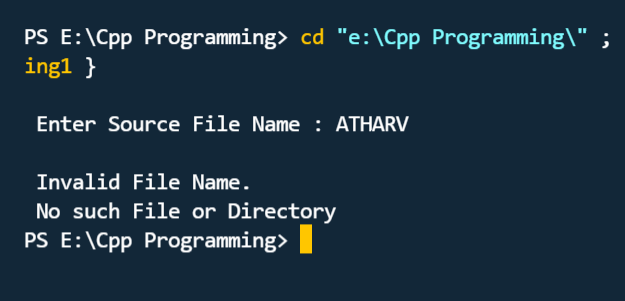
 cout<<"\n Total No. of Words : "<<now;

 cout<<"\n Total No. of Lines : "<<nol;

 return 0;

}

**OUTPUT:**



* Program: 26 (File Handling)

**Source Code:**

*//PROGRAM FOR handle multiple files and file operations  1904*

#include<iostream>

using namespace std;

#include<fstream>

 int main()

 {

*ofstream* fwrite("Alphabets.txt");

 fwrite<<"ABCDEFGHIJKLMNOPQRSTUVWXYZ";

fwrite.close();

*ifstream* fread("Alphabets.txt");

*ofstream* fwrite1("Vowels.txt");

*ofstream* fwrite2("Consonants.txt");

char c;

while(fread)

 {

fread.get(c);

if(c=='A' || c=='E' || c=='I' || c=='O' || c=='U')

 fwrite1<<c;

else

fwrite2<<c;

}

fread.close();

 fwrite1.close();

 fwrite2.close();

fread.open("Alphabets.txt");

*ifstream* fread1("Vowels.txt");

*ifstream* fread2("Consonants.txt");

cout<<"\n\nContents of Alphabets File\n";

 cout<<"--------------------------\n";

while(fread)

{

 fread.get(c);

 cout<<c;

}

fread.close();

cout<<"\n\nContents of Vowels File\n";

cout<<"-----------------------\n";

 while(fread1)

{

 fread1.get(c);

 cout<<c;

}

fread1.close();

cout<<"\n\nContents of Consonants File\n";

cout<<"---------------------------\n";

while(fread2)

{

fread2.get(c);

 cout<<c;

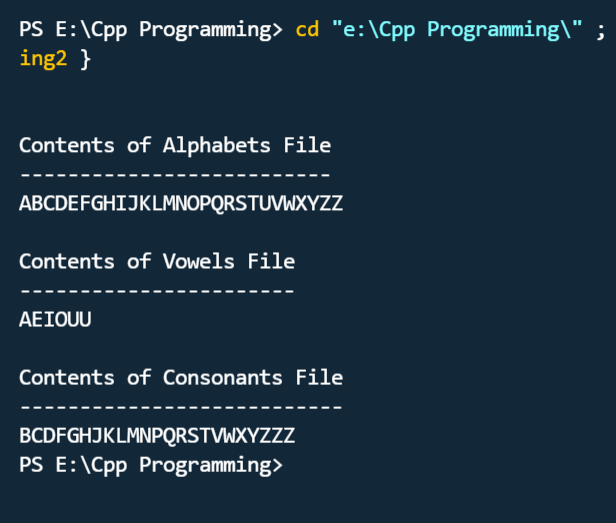
}

fread2.close();

return(0);

 }

**OUTPUT:**



* Program: 27 (File Handling)

**Source Code:**

*//Program for handle multiple files and file operations  1904*

#include<iostream>

#include<fstream>

 using namespace std;

class *student*

 {

char name[30];

 int age;

float percent;

public:

 void getdata()

 {

cout<<endl<<"Enter name: ";

cin>>name;

cout<<endl<<"Enter age: ";

cin>>age;

cout<<endl<<"Enter percentage: ";

cin>>percent;

 }

void showdata()

 {

cout<<endl<<name;

cout<<"\t\t"<< age;

 cout<<"\t\t"<<percent;

}

};

int main()

 {

*student* st;

*fstream* freadwrite("Student.txt", *ios*::ate | *ios*::in | *ios*::out);

 freadwrite.seekg(0,*ios*::beg);

cout<<endl<<"Current contents of file";

 while(freadwrite.read((char\*)&st,sizeof(st)))

 st.showdata();

freadwrite.clear();

cout<<endl<<"Enter details for one more student";

 st.getdata();

char c;

cin.get(c);

freadwrite.write((char\*)&st, sizeof(st));

 freadwrite.seekg(0);

cout<<endl<<"After addition of one more student";

 cout<<endl<<"Name \t\t Age \t\t Percentage";

while(freadwrite.read((char\*)&st, sizeof(st)))

 {

 st.showdata();

}

int n = freadwrite.tellg() / sizeof(st);

cout<<endl<<"Total no. of student record: ";

cout<<endl<<"Enter student number to be updated: ";

 int num;

 cin>>num;

 cin.get(c);

int l=(num-1) \* sizeof(st);

if(freadwrite.eof())

 freadwrite.clear();

freadwrite.seekp(l);

 cout<<endl<<"Enter new values for the student";

 st.getdata();

 cin.get(c);

freadwrite.write((char\*)&st, sizeof(st))<<flush;

freadwrite.seekg(0);

cout<<endl<<"After updation contents are";

cout<<endl<<"Name \t\t Age \t\t Percentage";

while(freadwrite.read((char\*)&st, sizeof(st)))

{

 st.showdata();

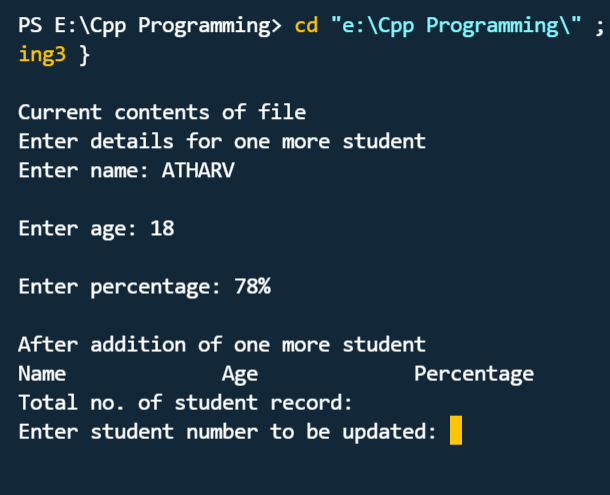
}

freadwrite.close();

 return(0);

 }

**OUTPUT:**



* Program: 28 (Template)

**Source Code:**

*//program for function template  1904*

#include<iostream>

using namespace std;

template <class *A*>

*A* sum(*A* x,*A* y)*//function template*

{

    return x+y;

}

int main()

{

    int a,b;

    float p,q;

    cout<<"enter two integer:";

    cin>>a>>b;

    cout<<"enter two float values:";

    cin>>p>>q;

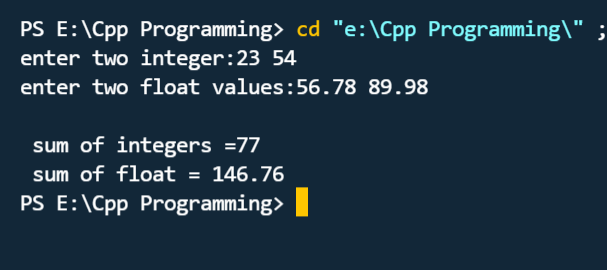
    cout<<"\n sum of integers ="<<sum(a,b);

    cout<<"\n sum of float = "<<sum(p,q);

    return 0;

}

**OUTPUT:**



* Program: 29 (Template)

**Source Code:**

*//PROGRAM FOR TEMPLATE CLASS LIBRARY  1904*

#include <iostream>

using namespace std;

template<class *A*>

class calculate

{ *A* x,y;

  public:

  void get()

  {

      cin>>x>>y;

  }

*A* add()

 { return x+y ;

 }

*A* sub()

{ return x-y ;

}

*A* multi()

{return x\*y;

}

void show()

{

    cout<<"\n addition is :"<<add();

    cout<<"\n subtraction is "<<sub();

    cout<<"\n multi is : "<< multi();

}

 };

int main()

{

    calculate<int> C;

    calculate<float> F;

    cout<<"\n enter 2 integers : ";

    C.get();

    cout<<"\n enter 2 decimal numbers :  ";

    F.get();

    cout<<"\n The answers of integer numbers are:  ";

    C.show();

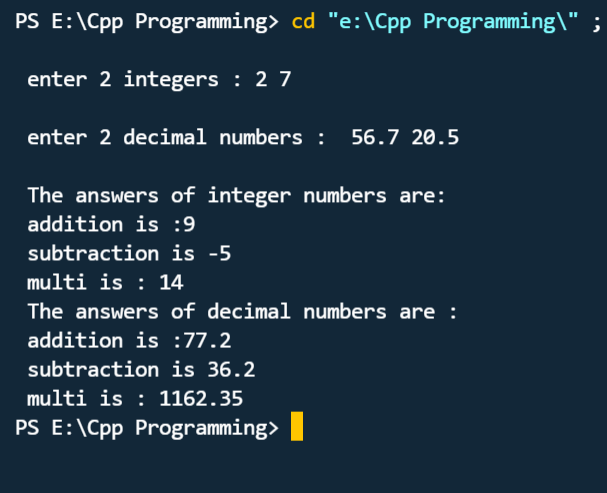
    cout<<"\n The answers of decimal numbers are :  ";

    F.show();

    return 0;

}

**OUTPUT:**



* Program: 29 (Template)

**Source Code:**

*//Program sorting ascending to descending  and vice-versa 1904*

#include<iostream>

using namespace std;

template <class *x*> void sort(*x* a[],int n)

{

    int i,j;

*x* temp;

    for(i=0;i<=n-2;i++)

    {

    for(j=0;j<=n-2;j++)

    {

        if(a[i]>a[j+1])

        {

            temp=a[i];

            a[i]=a[j+1];

            a[j+1]=temp;

        }

        }

    }

}

int main()

{

    int a[100],m,n,i;

    float b[100];

    cout<<"enter number of integers:";

    cin>>n;

    for(i=0;i<=n-1;i++)

    {

        cout<<"enter an integer:";

        cin>>a[i];

    }

    sort(a,n);

    cout<<"stored array:\n";

    for(i=0;i<=n-1;i++)

    {

        cout<<a[i]<<"\t";

    }

    cout<<endl;

    cout<<"enter number of float numbers:";

    cin>>m;

    for(i=0;i<=m-1;i++)

    {

        cout<<"enter a float number:";

        cin>>b[i];

    }

    sort(b,m);

    cout<<"stored array:\n";

    for(i=0;i<=m-1;i++)

    {

        cout<<b[i]<<"\t";

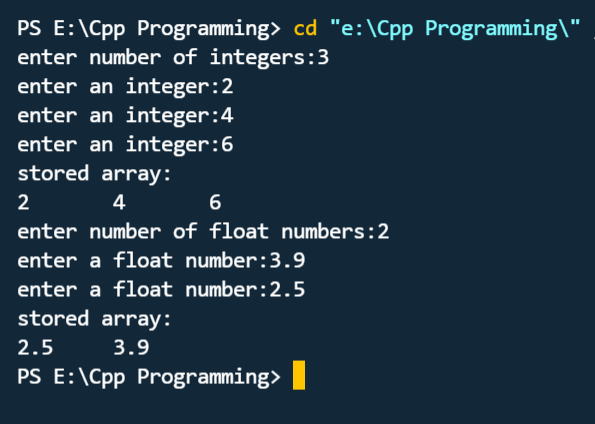
    }

    cout<<endl;

    return 0;

}

**OUTPUT:**



**\* \* \***