**CPP ASSIGNMENTS**

Assignment 01

Q1. write program to test Hello World.

#include<iostream>

usingnamespacestd;

intmain()

{

      cout<<"Hellow world"<<endl;

      return0;

}

Output :



---------------------------------------------------------------------------------------------------------

Q.2Write a program to adddition of two numbers .

#include<iostream>

usingnamespacestd;

intmain()

{

   intx,y;

   cout<<"Enter num1 : ";

   cin>>x;

   cout<<"Enter num2 : ";

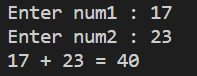
   cin>>y;

   cout<<x<<" + "<<y<<" = "<<x+y;

      return0;

}

Output :



---------------------------------------------------------------------------------------------------------

Q.3:Write a program to swap two numbers.

#include<iostream>

usingnamespacestd;

intmain()

{

   intx,y, temp;

   cout<<"Enter x : ";

   cin>>x;

   cout<<"Enter y : ";

   cin>>y;

   temp = x;

   x = y;

   y = temp;

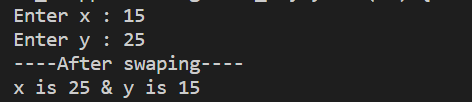
   cout<<"----After swaping----"<<endl;

   cout<<"x is "<<x<<" y is "<<y;

      return0;

}

Output :



---------------------------------------------------------------------------------------------------------

Q.4. Write a program to accept an integer and check if it is even or odd.

#include<iostream>

usingnamespacestd;

intmain()

{

    intn;

        cout<<"Enter the number";

        cin>>n;

        if(n%2==0)

        {

            cout<<"it is even";

        }

        else

        {

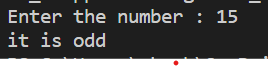
            cout<<"it is odd";

        }

      return0;

}

Output :

---------------------------------------------------------------------------------------------------------

Q.5. Write a program to accept a number and check if it is divisible by 5 and 7.

#include<iostream>

usingnamespacestd;

intmain()

{

    intn;

         cout<<"Enter the number ";

         cin>>n;

         if(n%7==0&&n%5==0)

         {

            cout<<"it is divisible by "<<7<<" & "<<5;

         }

         elseif (n%5==0)

         {

            cout<<"it is divisible by "<<5;

         }

         elseif (n%7==0)

         {

            cout<<"it is divisible by "<<7;

         }

         else

         {

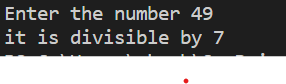
            cout<<"It is not divisible by 5 & 7";

         }

}

Output :





---------------------------------------------------------------------------------------------------------

6. Write a program, which accepts annual basic salary of an employee and calculates and displays theIncome tax as per the following rules. Basic: < 1, 50,000 Tax = 0

1, 50,000 to 3,00,000 Tax = 20%

> 3,00,000 Tax = 30%

#include<iostream>

usingnamespacestd;

intmain()

{

    intn;

    cout<<"Enter the basic salray\n";

    cin>>n;

    if(n<=150000)

    {

        cout<<"Income Tax is "<<0;

    }

    elseif(n>150000&&300000>n)

    {

        cout<<"Income tax is 20% = "<<n\*0.2;

    }

    elseif (300000<=n)

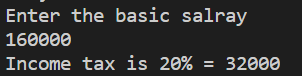
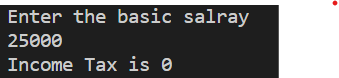
    {

        cout<<"Income tax is 30% = "<<n\*0.3;

    }

}

Output :



---------------------------------------------------------------------------------------------------------

7. Accept a lowercase character from the user and check whether the character is a vowel or consonant.

(Hint: a, e, i, o, u are vowels)

#include<iostream>

usingnamespacestd;

intmain()

{

    charch;

    cout<<"Enter the char : ";

    cin>>ch;

    if(ch == 'a'||ch == 'e'|| ch == 'i'|| ch == 'o'|| ch =='u')

    {

        cout<<"this is voval";

    }

    else

    {

        cout<<"This is consonant ";

    }

}

Output :



---------------------------------------------------------------------------------------------------------

Q.8. Write a program to input angles of a triangle and check whether triangle is valid or not.

#include<iostream>

usingnamespacestd;

intmain()

{

    inta,b,c;

        cout<<"Enter the angles"<<endl;

        cin>>a>>b>>c;

        if((a+b+c)==180)

        {

            cout<<"This is valid trangle";

        }

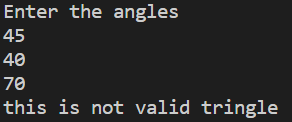
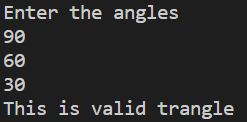
        else

        {

            cout<<"this is not valid tringle";

        }

}



---------------------------------------------------------------------------------------------------------

Q.9Write a program to find factorial of a given number. ex:no5 fact=5\*4\*3\*2\*1=120

#include<iostream>

usingnamespacestd;

intmain()

{

     intn,fact = 1;

        cout<<"enter the no : ";

        cin>>n;

        for (inti = 1; i<= n; i++)

        {

            fact = fact\*i;

        }

        cout<<"Factorial of "<<n<<" is "<<fact;

}

Output :



---------------------------------------------------------------------------------------------------------

Q.10Write a program to find m to the power n. m=3 and n=4 so 3\*3\*3\*3

#include<iostream>

usingnamespacestd;

intmain()

{

    intn,m,pow;

        cout<<"Enter the number : ";

        cin>>n;

        cout<<"Enter the power : ";

        cin>>m;

        pow = n;

        for (inti = 1; i<m; i++)

        {

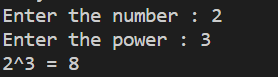
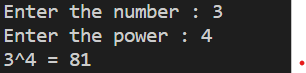
            pow = pow \* n;

        }

        cout<<n<<"^"<<m<<" = "<<pow;

}

Output :



---------------------------------------------------------------------------------------------------------

Q.11Check if number is a prime number or not

#include<iostream>

usingnamespacestd;

intmain()

{

     intn;

        cout<<"Enter the no : ";

        cin>>n;

        boolb = true;

        for (inti = 2; i<n-1; i++)

        {

            if(n%i==0)

            {

               b = false;

                break;

            }

            else

            {

               b = true;

            }

        }

        if(b == true)

        {

            cout<<"This is prime no ";

        }

        else

        {

            cout<<"This is not prime no ";

        }

}



---------------------------------------------------------------------------------------------------------

Q.12 Sum of series

#include<iostream>

usingnamespacestd;

intmain()

{

        intsum = 0, n;

        cout<<"Enter the number : ";

        cin>>n;

        for (inti = 0; i<= n; i++)

        {

            sum = sum + i;

        }

        cout<<"The sum is "<<sum;

}

Output :

---------------------------------------------------------------------------------------------------------

Q.13Check whether the number is palindrome or not?

#include<iostream>

usingnamespacestd;

intmain()

{

           intorignal, remainder, n,revers = 0;

       cout<<"Enter the number :";

       cin>>n;

       orignal = n;

       while (n!=0)

       {

           remainder = n%10;

           revers = revers \* 10 +remainder;

           n = n/10;

       }

       if (orignal == revers)

       {

           cout<<"This is palindrome";

       }

       else

       cout<<"This is not palindrome";

}

Output :

---------------------------------------------------------------------------------------------------------

Q.14:Write a program to find sum of all even and odd numbers between 1 to n.

#include<iostream>

usingnamespacestd;

intmain()

{

         int  n,even = 0, odd = 0;

        cout<<"Enter the number from 1 to :";

        cin>>n;

        for (inti = 0; i<n; i++)

        {

            if(i%2==0)

            {

                even = even + i;

            }

            else

            {

                odd = odd + i;

            }

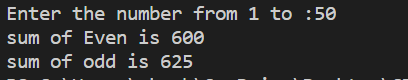
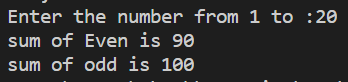
        }

        cout<<"sum of Even is "<<even<<endl;

        cout<<"sum of odd is "<<odd<<endl;

}

Output :

---------------------------------------------------------------------------------------------------------

15: Write a program to enter a number and print its reverse.

#include<iostream>

usingnamespacestd;

intmain()

{

       intn, revers = 0, remainder = 0;

        cout<<"ENter the number"<<endl;

        cin>>n;

        while (n!=0)

        {

            remainder =  n%10;

            revers = revers \* 10 +remainder;

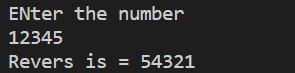
            n = n/10;

        }

        cout<<"Revers is = "<<revers;

}

Output :



---------------------------------------------------------------------------------------------------------

16:Write a program to print all Prime numbers between 1 to n.

#include<iostream>

usingnamespacestd;

intmain()

{

        intn;

        boolflag = false;

         cout<<"Enter the number :";

         cin>>n;

         for (inti = 2; i<n; i++)

         {  boolflag = true;

            for (intj = 2; j<i; j++)

            {

                if(i%j==0)

                {

                    flag = false;

                }

            }

            if(flag == true)

            {

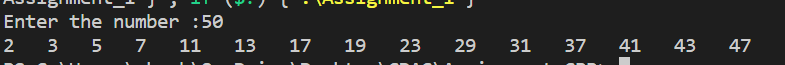
                cout<<i<<"   ";

            }

         }

}

Output :



---------------------------------------------------------------------------------------------------------

17:Write a program to check entered number is Armstrong number or not.

#include<iostream>

usingnamespacestd;

intmain()

{

       intn,p,remainder, sum = 0,count = 0,orignal;

            cout<<"Enter the number :";

            cin>>n;

            p = n;

            orignal = n;

            while (p!=0)

            {

                count++;

                p/=10;

            }

            while (n!=0)

            {

                intx = 1;

                remainder = n%10;

                for (inti = 0; i<count; i++)

                {

                    x = x \* remainder;

                }

                sum = sum + x;

                n/=10;

            }

            if (sum == orignal)

            {

                cout<<"This is Armstrong number !";

            }

            else

            {

                cout<<"This is not armstrong number !";

            }

}

Output :



---------------------------------------------------------------------------------------------------------

18:Write a program to find greatest of three numbers using nested if-else.

#include<iostream>

usingnamespacestd;

intmain()

{

        inta,b,c;

        cout<<"Enter three number";

        cin>>a>>b>>c;

        if (a>b)

        {

            if(a>c)

            {

                cout<<"Gretest no is = "<<a;

            }

            else

            {

                cout<<"Gretest no is = "<<c;

            }

        }

        else

        {

            if(b>c)

            {

                cout<<"Gretest no is = "<<b;

            }

            else

            {

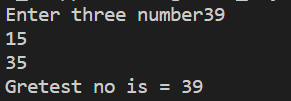
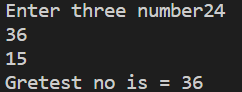
                cout<<"Gretest no is = "<<c;

            }

        }

}

Output :



---------------------------------------------------------------------------------------------------------

19:Create menu driven program for Pizza Shop.And display total amount,

#include<iostream>

usingnamespacestd;

intmain()

{int choice, qty, total = 0;

            cout<<"Enter the choice\n1 for tomato 100\n2 for cheese 120\n3 for double cheese 160\n4 for oniun 50\n5 for total\n";

        do

        {

            cout<<"Order : \n";

            cin>>choice;

            switch (choice)

            {

            case1:

                cout<<"Enter the Qty of Tomato Pizza : ";

                cin>>qty;

                total = total + qty \* 100;

                break;

            case2:

                cout<<"Enter the Qty of cheese Pizza : ";

                cin>>qty;

                total = total + qty \* 120;

                break;

            case3:

                cout<<"Enter the Qty of double cheese Pizza : ";

                cin>>qty;

                total = total + qty \* 160;

                break;

            case4:

                cout<<"Enter the Qty of oniun Pizza : ";

                cin>>qty;

                total = total + qty \* 50;

                break;

            case5:

                cout<<"Your Total is "<<total;

                break;

            default:

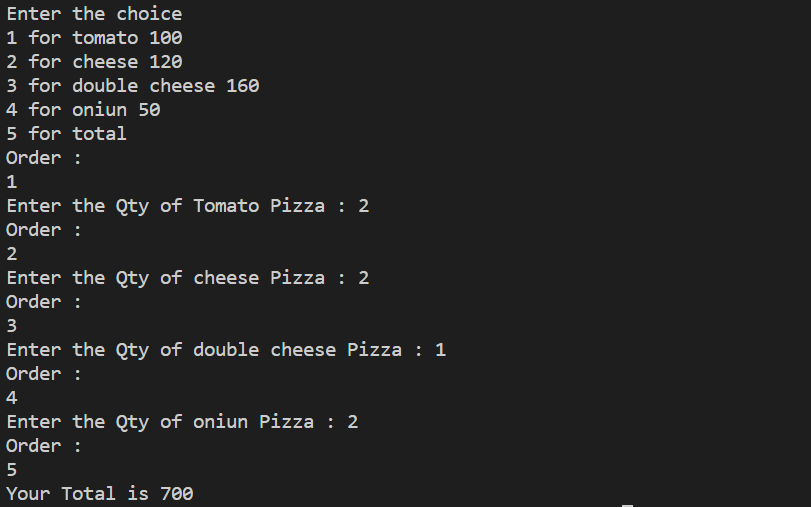
                cout<<"enter valied choise";

                break;

            }

        }while(choice!=5);

Output:



---------------------------------------------------------------------------------------------------------

20:Accept a single digit from the user and display it in words. For example, if digit entered is 9, display Nine.

#include<iostream>

usingnamespacestd;

intmain()

{ intn;

        cout<<"Enter the number from 1 to 9 :";

        cin>>n;

        switch (n)

        {

        case1:

            cout<<"one";

            break;

        case2:

            cout<<"two";

            break;

        case3:

            cout<<"three";

            break;

        case4:

            cout<<"four";

            break;

        case5:

            cout<<"five";

            break;

        case9:

            cout<<"nine";

            break;

        case6:

            cout<<"six";

            break;

        case7:

            cout<<"seven";

            break;

        case8:

            cout<<"eight";

            break;

        default:

        cout<<"Enter valid no \n";

            break;

        }

}

Output :





---------------------------------------------------------------------------------------------------------

21. Write a program, which accepts two integers and an operator as a character (+ - \* / ), performs the

corresponding operation and displays the result.

#include<iostream>

usingnamespacestd;

intmain()

{

        intn,m;

        cout<<"Enter two numbers ";

        cin>>m>>n;

        charch;

        cout<<"Enter + , - , \*, /\n";

        cin>>ch;

        switch (ch)

        {

        case'+':

            cout<<n<<" + "<<m<<" = "<<m+n;

            break;

         case'-':

            cout<<n<<" - "<<m<<" = "<<m-n;

            break;

         case'\*':

            cout<<n<<" \* "<<m<<" = "<<m\*n;

            break;

         case'/':

            cout<<n<<" / "<<m<<" = "<<m/n;

            break;

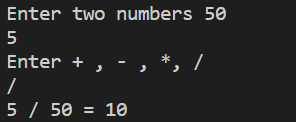
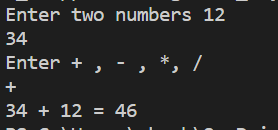
        default:

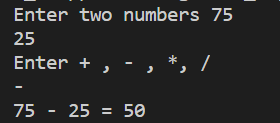
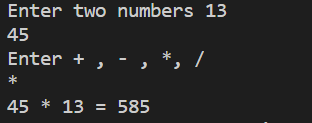
            break;

        }

}

Output :





---------------------------------------------------------------------------------------------------------

Assignment 02

---------------------------------------------------------------------------------------------------------

1:Write a program that accepts numbers continuously as long as the number is positive and prints the sum of the given numbers.

#include<iostream>

usingnamespacestd;

intmain()

{

     intn,sum = 0;

     do

     {

         cout<<"Enter number =";

         cin>>n;

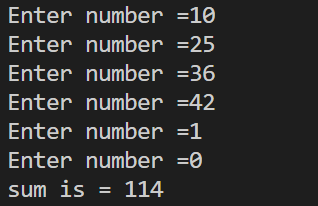
         sum = sum + n;

     }while (n!=0);

     cout<<"sum is = "<<sum;

}

Output :



---------------------------------------------------------------------------------------------------------

2. Write a program to accept two integers x and n and compute x raised to n.

#include<iostream>

usingnamespacestd;

intmain()

{

     intx,n,sum=1;

        cout<<"Enter the x number : ";

        cin>>x;

        cout<<"Enter the n number : ";

        cin>>n;

        for (inti = 0; i<n; i++)

        {

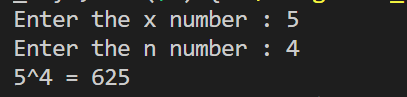
            sum = sum \* x;

        }

        cout<<x<<"^"<<n<<" = "<<sum;

}

Output :



---------------------------------------------------------------------------------------------------------

3. Write a program to accept a character, an integer n and display the next n characters.

#include<iostream>

usingnamespacestd;

intmain()

{

     charc;

        intn;

        cout<<"enter the char :";

        cin>>c;

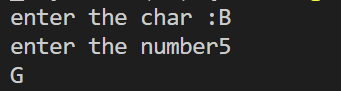
        cout<<"enter the number";

        cin>>n;

        cout<<char(c+n);

}

Output :



--------------------------------------------------------------------------------------------------------

4. Write a program to calculate factorial of a number.

#include<iostream>

usingnamespacestd;

intmain()

{

    intn;

        cout<<"Enter the number";

        cin>>n;

        intsum = 1;

        for (inti = n; i>0; i--)

        {

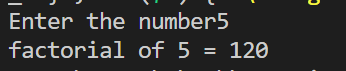
            sum = sum\*i;

        }

        cout<<"factorial of "<<n<<" = "<<sum;

}

Output :



---------------------------------------------------------------------------------------------------------

5. Write a program to calculate factors of a given number.

#include<iostream>

usingnamespacestd;

intmain()

{

    intn;

        cout<<"Enter the number :";

        cin>>n;

        cout<<"Factor of "<<n<<" = ";

        for (inti = 1; i<n; i++)

        {

            if(n%i==0)

            {

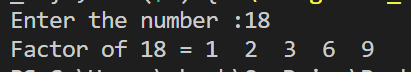
                cout<<i<<"  ";

            }

        }

}

Output :



---------------------------------------------------------------------------------------------------------

6. Accept two numbers and calculate GCD of them.

#include<iostream>

usingnamespacestd;

intmain()

{

  intnum1,num2,gcd;

        cout<<"Enter the num1 and num 2 : ";

        cin>>num1>>num2;

        for (inti = 1; i<= num1&&i<= num2; i++)

        {

            if (num1%i==0&&num2%i==0)

            {

                gcd = i;

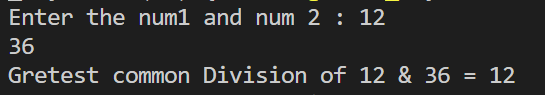
            }

        }

        cout<<"Gretest common Division of "<<num1<<" & "<<num2<<" = "<<gcd;

        }

Output :



---------------------------------------------------------------------------------------------------------

7. Write a menu driven program to do following operations :

a) Compute area of circleb) Compute area of rectanglec) Compute area of triangled) Exit

Display menu, ask choice to the user, depending on choice accept the parameters and perform the

operation. Continue this process until user selects exit option.

#include<iostream>

usingnamespacestd;

intmain()

{

 intchoise;

    floatx,y,n;

    cout<<"\nEnter your choise\n1 for area of circle\n2 for area of rectangle\n3 for area of triangle\n4 for EXIT\n";

    do

    {

        cout<<"\nEnter choise : ";

        cin>>choise;

        switch (choise)

        {

        case1:

            cout<<"Enter the redious of circle : ";

            cin>>n;

            cout<<"Area of circle is = "<<(3.14\*n\*n);

            break;

         case2:

            cout<<"Enter the lenght : ";

            cin>>x;

            cout<<"Enter the breadth : ";

            cin>>y;

            cout<<"Area of rectangle is = "<<x\*y;

            break;

        case3:

            cout<<"Enter the hight : ";

            cin>>x;

            cout<<"Enter the breadth : ";

            cin>>y;

            cout<<"Area of triangle is ="<<(x\*y)/2;

            break;

        case4:

            cout<<"Exiting from loop\n";

            break;

        default:

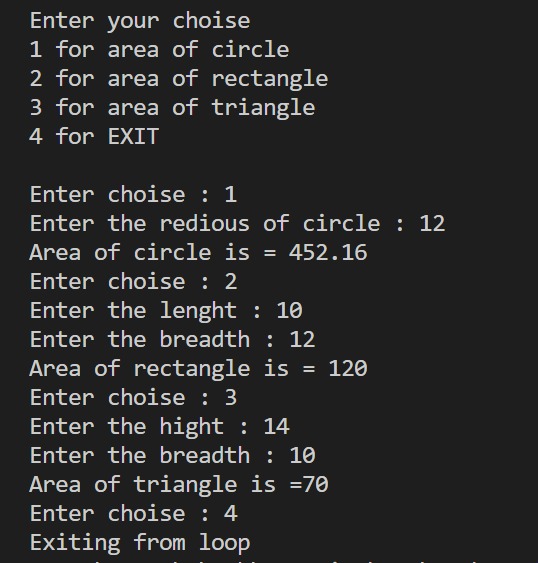
            printf("enter the valid choise\n");

        }

    } while (choise != 4);

}

Output :



---------------------------------------------------------------------------------------------------------

8. Write a program to print all prime numbers between 1 to n

#include<iostream>

usingnamespacestd;

intmain()

{

intn;

    cout<<"Enter the number :";

    cin>>n;

    boolflag = true;

    for (inti = 2; i<n; i++)

    {

        flag = true;

        for (intj = 2; j<i; j++)

        {

            if (i % j == 0)

            {

                flag = false;

                break;

            }

        }

        if (flag == true)

        {

            cout<<i<<"  ";

        }

    }

}

Output :



---------------------------------------------------------------------------------------------------------

Assignment 3

1:Write a program to create student class with data members rollno, marks1,mark2,mark3.

Accept data (acceptInfo()) and display using display member function.

Also display total,percentage and grade.

#include<iostream>

usingnamespacestd;

classstudent

{

    introll;

    intmarks[3];

    intpersentage;

    public:

    voidaccept\_data()

    {

        cout<<"Enter roll no :";

        cin>>roll;

        for (inti = 0; i<3; i++)

        {

            cout<<"Enter the marks "<<i+1<<" : ";

            cin>>marks[i];

        }

    }

    voiddisplay\_data()

    {

        intpersentage;

        cout<<"roll no = "<<roll;

        for (inti = 0; i<3; i++)

        {

            cout<<"\nMarks "<<i+1<<" = "<<marks[i]<<"  ";

        }

        persentage = ((marks[0]+marks[1]+marks[2])\*100)/300;

        cout<<"\npersentage is = "<<persentage<<"%";

        if (persentage<35)

        {

            cout<<"\nGrade is Fail";

        }

        elseif (persentage>35&&persentage<80)

        {

            cout<<"\nGrade is A";

        }

        elseif (persentage>80&&persentage<=100)

        {

            cout<<"\nGrade is A+";

        }

    }

};

intmain()

{

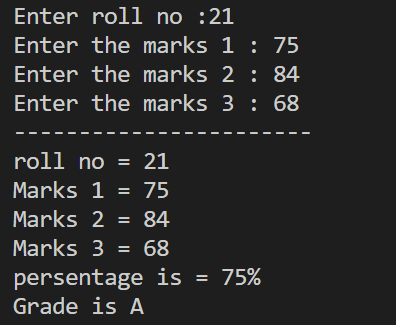
    studentst;

    st.accept\_data();

    st.display\_data();

}

Output :



---------------------------------------------------------------------------------------------------------

1. Create a class Person with data members as name, age, city.

members. Also add accept and display function. . Create the

object of this class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classperson

{

    stringname,city;

    intage;

    public:

        voidaccept\_data()

        {

            cout<<"Enter the name : ";

            cin>>name;

            cout<<"Enter the age : ";

            cin>>age;

            cout<<"Enter the city : ";

            cin>>city;

        }

        voiddispaly()

        {

            cout<<"Name is = "<<name<<endl;

            cout<<"Age is = "<<age<<endl;

            cout<<"City is = "<<city<<endl;

        }

};

intmain()

{

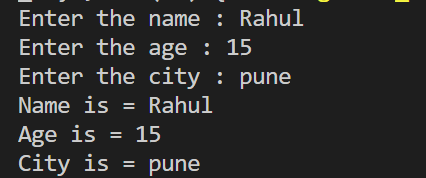
    personp;

    p.accept\_data();

    p.dispaly();

}

Output :



---------------------------------------------------------------------------------------------------------

2. Create a class Date with data members as dd, mm, yy. Create AcceptDate function. Also add the display function. Create the

object of this class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classdate

{

    intdd,mm,yy;

    public:

    voidaccept()

    {

        cout<<"Enter dd :";

        cin>>dd;

        cout<<"Enter mm :";

        cin>>mm;

        cout<<"Enter yy :";

        cin>>yy;

    }

    voiddisplay()

    {

        cout<<dd<<" : "<<mm<<" : "<<yy<<endl;

    }

};

intmain()

{

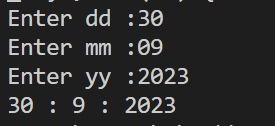
    dated;

    d.accept();

    d.display();

}

Output :



---------------------------------------------------------------------------------------------------------

3. Create a class Book with data members as bname,id,author,price. Write AcceptBook function . Also add the display function. Create Default and Parameterized constructors. Create

the object of this class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classbook

{

    stringbname;

    stringauthor;

    intid,price;

    public:

    voidaccept\_data()

    {

        cout<<"Enter the id : ";

        cin>>id;

        cout<<"Price : ";

        cin>>price;

        cout<<"Enter book name : ";

        cin>>bname;

        cout<<"Enter the author name : ";

        cin>>author;

        cout<<endl;

    }

    voiddisplay()

    {

        cout<<"book name "<<bname<<endl;

        cout<<"book author "<<author<<endl;

        cout<<"book id "<<id<<endl;

        cout<<"book price "<<price<<endl;

    }

};

intmain()

{

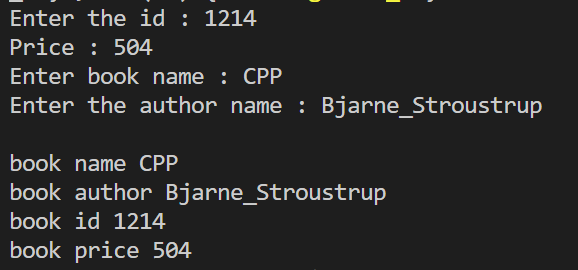
    bookb;

    b.accept\_data();

    b.display();

}

Output :



---------------------------------------------------------------------------------------------------------

4. Create a class Point with data members as x,y. Add AcceptPoint and add the display function. Create the object of this

class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classPoint

{

    intx,y;

    public:

    voidaccept()

    {

        cout<<"Enter the x ";

        cin>>x;

        cout<<"Enter the y ";

        cin>>y;

    }

    voiddisplay()

    {

        cout<<"X is = "<<x<<endl;

        cout<<"y is = "<<y<<endl;

    }

};

intmain()

{

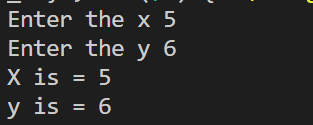
    Pointp;

    p.accept();

    p.display();

}

Output :



---------------------------------------------------------------------------------------------------------

5. Create a class ComplexNumber with data members real, imaginary. Create AcceptComplexNumber() and the display function. Create the object of this class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classcomplex {

public:

    intreal;

    intimaginary;

    complex()

    {

        real = 0;

        imaginary = 0;

    }

    complex(intr, inti)

    {

        real = r;

        imaginary = i;

    }

    complexAcceptComplexNumber(complexC1, complexC2)

    {

        complexres;

        res.real = C1.real + C2.real;

        res.imaginary = C1.imaginary + C2.imaginary;

        returnres;

    }

};

intmain()

{

    complexc1(8, 5);

    cout<<c1.real<<" + i"<<c1.imaginary<<endl;

    complexc2(7, 9);

    cout<<c2.real<<" + i"<<c2.imaginary<<endl;

    complexc3;

    c3=c3.AcceptComplexNumber(c1, c2);

    cout<<c3.real<<" + i"<<c3.imaginary;

    cout<<"\n\n";

    complexA(2, 7);

    cout<<A.real<<" + i"<<A.imaginary<<endl;

    complexB(10, 6);

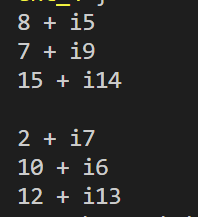
    cout<<B.real<<" + i"<<B.imaginary<<endl;

    complexC;

    C=C.AcceptComplexNumber(A, B);

    cout<<C.real<<" + i"<<C.imaginary;

output :



---------------------------------------------------------------------------------------------------------

Assignment 04

1:Write a program to create student class with data members rollno, marks1,mark2,mark3.

Accept data (acceptInfo()) and display using display member function.

Also display total,percentage and grade.

#include<iostream>

usingnamespacestd;

classstudent

{

    introll;

    intmarks[3];

    intpersentage;

    public:

    voidaccept\_data()

    {

        cout<<"Enter roll no :";

        cin>>roll;

        for (inti = 0; i<3; i++)

        {

            cout<<"Enter the marks "<<i+1<<" : ";

            cin>>marks[i];

        }

    }

    voiddisplay\_data()

    {

        intpersentage;

        cout<<"roll no = "<<roll;

        for (inti = 0; i<3; i++)

        {

            cout<<"\nMarks "<<i+1<<" = "<<marks[i]<<"  ";

        }

        persentage = ((marks[0]+marks[1]+marks[2])\*100)/300;

        cout<<"\npersentage is = "<<persentage<<"%";

        if (persentage<35)

        {

            cout<<"\nGrade is Fail";

        }

        elseif (persentage>35&&persentage<80)

        {

            cout<<"\nGrade is A";

        }

        elseif (persentage>80&&persentage<=100)

        {

            cout<<"\nGrade is A+";

        }

    }

};

intmain()

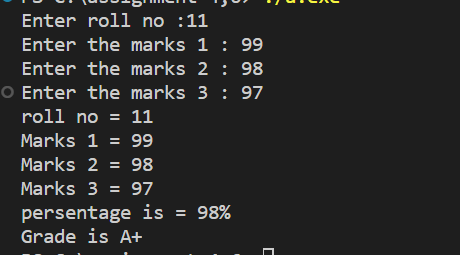
{

    studentst;

    st.accept\_data();

    st.display\_data();

}



----------------------------------------------------------------------------------------------------------------------------------

2. Create a class Date with data members as dd, mm, yy. Write getters and setters for all the data members. Also add the display function. Create Default and Parameterized constructors. Create the

object of this class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classPerson

{

    stringname;

    intage;

    stringcity;

    public:

    Person()

    {

        cout<<"default constructor call"<<endl;

        name="Akash";

        age = 25;

        city="Solapur";

    }

    Person(intage, stringname, stringcity)

    {

        cout<<"Parameterized constructor call"<<endl;

        this->age = age;

        this->name=name;

        this->city=city;

    }

    voidaccept\_data()

    {

        cout<<"Enter the name : ";

        cin>>name;

        cout<<"Enter the age : ";

        cin>>age;

        cout<<"Enter the city : ";

        cin>>city;

        cout<<endl;

    }

    voiddisplay()

    {

        cout<<"Name is = "<<name<<endl;

        cout<<"Age is = "<<age<<endl;

        cout<<"City = "<<city<<endl;

    }

    voidsetname(stringname)

    {

        cout<<"name is updated"<<endl;

        this->name=name;

    }

    voidsetage(intage)

    {

        cout<<"Age is updated"<<endl;

        this->age = age;

    }

    voidsetcity(stringcity)

    {

        cout<<"City is updated"<<endl;

        this->city=city;

    }

    stringgetname()

    {

        returnname;

    }

     intgetage()

    {

        returnage;

    }

     stringgetcity()

    {

        returncity;

    }

};

intmain()

{

    Personp;

    Person(26,"rahul","mumbai");

    cout<<endl;

    p.accept\_data();

    p.display();

    cout<<endl;

    p.setage(27);

    p.setcity("jalgaon");

    p.setname("jayash");

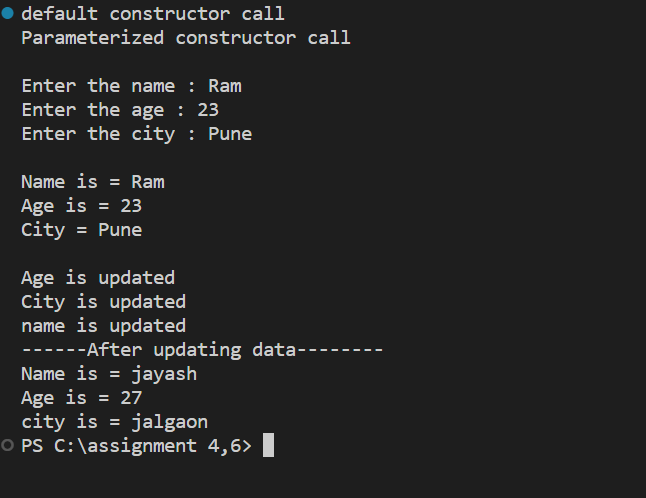
    cout<<"------After updating data--------"<<endl;

    cout<<"Name is = "<<p.getname()<<endl;

    cout<<"Age is = "<<p.getage()<<endl;

    cout<<"city is = "<<p.getcity()<<endl;

}



----------------------------------------------------------------------------------------------------------------------------------

3. Create a class Book with data members as bname,id,author,price. Write getters and setters for all the

data members. Also add the display function. Create Default and Parameterized constructors. Create

the object of this class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classData

{

    intdd,mm,yy;

    public:

    Data()

    {

        cout<<"Default constructor call"<<endl;

        dd = 01; mm = 06; yy = 2023;

    }

    Data(intdd, intmm, intyy)

    {

        cout<<"Parameterized constructor call"<<endl;

        this->dd = dd;

        this->mm = mm;

        this->yy = yy;

    }

    voidaccept\_data()

    {

        cout<<"Enter dd : ";

        cin>>dd;

        cout<<"Enter mm : ";

        cin>>mm;

        cout<<"Enter yy : ";

        cin>>yy;

        cout<<endl;

    }

    voiddisplay\_data()

    {

        cout<<"Data is = "<<dd<<" : "<<mm<<" : "<<yy<<endl;

    }

    voidsetdd(intdd)

    {

        this->dd = dd;

    }

    voidsetmm(intmm)

    {

        this->mm = mm;

    }

    voidsetyy(intyy)

    {

        this->yy = yy;

    }

    intgetdd()

    {

        returndd;

    }

    intgetmm()

    {

        returnmm;

    }

    intgetyy()

    {

        returnyy;

    }

};

intmain()

{

    Datad;

    Data(01,01,2023);

    d.accept\_data();

    d.display\_data();

    d.setdd(28);

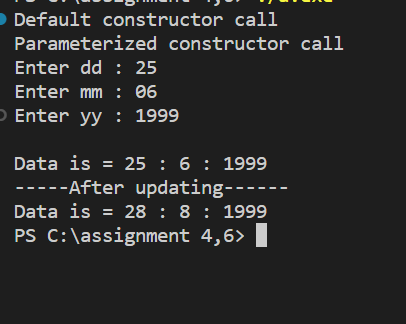
    d.setmm(8);

    d.setyy(1999);

    cout<<"-----After updating------"<<endl;

    d.display\_data();

}



----------------------------------------------------------------------------------------------------------------------------------

4. Create a class Point with data members as x,y. Create Default and Parameterized constructors. Write

getters and setters for all the data members. Also add the display function. Create the object of this

class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classbook

{

    stringbname;

    stringauthor;

    intid,price;

    public:

    book()

    {

        cout<<"defult constructor call"<<endl;

        bname="Ramayan";

        author="valmiki";

        id = 101;

        price = 270;

    }

    book(stringname,stringauthor, intid,intprice)

    {

        cout<<"Parameterized constructor call "<<endl;

        this->bname=name;

        this->author=author;

        this->id = id;

        this->price;

    }

    voidaccept\_data()

    {

        cout<<"Enter the id : ";

        cin>>id;

        cin.ignore(1,'\n');

        cout<<"Enter book name : ";

        getline(cin,bname);

        cout<<"Enter the author name : ";

        cin>>author;

        cout<<"Price : ";

        cin>>price;

        cout<<endl;

    }

    voiddisplay()

    {

        cout<<"book name : "<<bname<<endl;

        cout<<"book author : "<<author<<endl;

        cout<<"book id : "<<id<<endl;

        cout<<"book price : "<<price<<endl;

    }

    voidsetbname(stringbname)

    {

        cout<<"Updated book name"<<endl;

        this->bname=bname;

    }

    stringgetname()

    {

        returnbname;

    }

    voidsetauthor(stringauthor)

    {

        cout<<"Updated book author"<<endl;

        this->author=author;

    }

    stringgetauthor()

    {

        returnauthor;

    }

    voidsetid(intid, intprice)

    {

        cout<<"Updated book id and price"<<endl;

        this->id = id;

        this->price = price;

    }

    intgetid()

    {

        returnid;

    }

    intgetprice()

    {

        returnprice;

    }

};

intmain()

{

    bookb;

    book("chanakya niti","chanakya",102,300);

    b.accept\_data();

    cout<<endl;

    b.display();

    cout<<endl;

    b.setbname("C++");

    b.setauthor("Stroustrup");

    b.setid(1989, 500);

    cout<<endl;

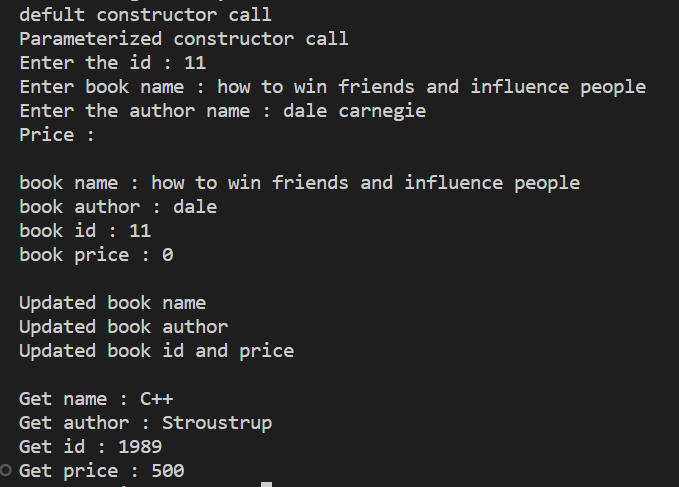
    cout<<"Get name : "<<b.getname()<<endl;

    cout<<"Get author : "<<b.getauthor()<<endl;

    cout<<"Get id : "<<b.getid()<<endl;

    cout<<"Get price : "<<b.getprice()<<endl;

}



----------------------------------------------------------------------------------------------------------------------------------

5. Create a class ComplexNumber with data members real, imaginary. Create Default and Parameterized constructors. Write getters and setters for all the data members. Also add the display function. Create the object of this class in main method and invoke all the methods in that class.

#include<iostream>

usingnamespacestd;

classPoint

{

    intx,y;

    public:

    Point()

    {

        cout<<"Default Constructor call "<<endl;

        x = 10; y = 20;

    }

    Point(intx,inty)

    {

        cout<<"Paramerized constructor call"<<endl;

        this->x = x;

        this->y = y;

    }

    voidaccept()

    {

        cout<<"Enter the x ";

        cin>>x;

        cout<<"Enter the y ";

        cin>>y;

    }

    voiddisplay()

    {

        cout<<"X is = "<<x<<endl;

        cout<<"y is = "<<y<<endl;

    }

    voidsetx(intx)

    {

        this->x = x;

    }

    voidsety(inty)

    {

        this->y = y;

    }

    intgetx()

    {

        returnx;

    }

    intgety()

    {

        returny;

    }

};

intmain()

{

    Pointp;

    p.accept();

    p.display();

    p.setx(15);

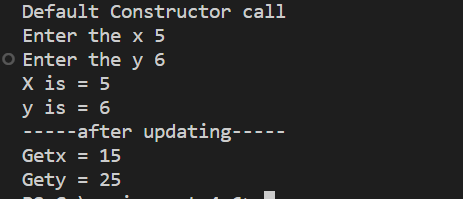
    p.sety(25);

    cout<<"-----after updating-----"<<endl;

    cout<<"Getx = "<<p.getx()<<endl;

    cout<<"Gety = "<<p.gety()<<endl;

}



----------------------------------------------------------------------------------------------------------------------------------

INHERITANCE

Animal Hierarchy:

Problem Statement: Create a hierarchy of animal classes. Start with a base class Animal and then create derived classes like Mammal, Bird, and Fish. Each of these derived classes should have specific properties and methods related to their respective categories of animals.

#include<iostream>

using namespace std;

class animal{

private: string place;

public:

   void accept(){

    cout<<"place"<<endl;

    cin>>place;

   }

virtual void spfun(){

cout<<"I am animals"<<endl;

}

virtual void dispaly(){

    cout<<place<<" "<<endl;

    spfun();

}

};

class mammals: public animal{

private:

 string place="ambipours";

 public:

 void spfun(){

    cout<<"i am mammal"<<endl;

 }

 void dispaly(){

    animal::dispaly();

 }

};

class bird: public animal{

private: string place="air";

public:

void spfun(){

    cout<<"I am bird"<<endl;

}

void dispaly(){

    animal::dispaly();

}

};

class fish: public animal{

private: string place="water";

public:

 void spfun(){

    cout<<"I am fish"<<endl;

 }

 void dispaly(){

    animal::dispaly();

 }

};

int main(){

 fish f1;

 f1.dispaly();

 f1.spfun();

 mammals m1;

 m1.dispaly();

 m1.spfun();

 bird b1;

b1.dispaly();

b1.spfun();

animal \*ptr[4];

ptr[0]=&f1;

ptr[1]=&m1;

ptr[2]=&b1;

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

{

    cout<<"---------"<<endl;

    ptr[i]->dispaly();

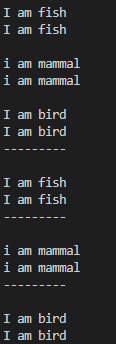
    ptr[i]->spfun();

}

    return 0 ;

}

Op=



----------------------------------------------------------------------------------------------------------------------------------

Shape Hierarchy:

Problem Statement: Design a hierarchy of shape classes. Begin with a base class Shape and then create derived classes like Circle, Rectangle, and Triangle. Each shape should have methods for calculating area and perimeter specific to its geometry.

#include<iostream>

using namespace std;

class shape{

public:

virtual void accept()=0;

virtual void calculate()=0;

};

class rectangle:public shape{

int l,b;

public:

void accept(){

cout<<"Enter l="<<endl;

cin>>l;

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

cin>>b;

}

void calculate(){

    cout<<"Area of rectangle="<<(l\*b)<<endl;

}

};

class circle:public shape{

private:int r;

public:

void accept(){

    cout<<"Enter radius="<<endl;

    cin>>r;

}

void calculate(){

    cout<<"Area of circle="<<(3.14\*r\*r)<<endl;

}

};

class triangle: public shape{

private: int b,h;

public:

void accept(){

    cout<<"Enter b  and h="<<b<<h<<endl;

    cin>>h;

    cin>>b;

}

void calculate(){

    cout<<"area of triangle="<<((b\*h)/2)<<endl;

}

};

int main(){

int ch;

do

{

cout<<"enter choice  1.rectangle  \n 2.circle \n 3.triangle \n 4.exit"<<endl;

cin>>ch;

switch (ch)

{

case 1:

     rectangle r1;

    r1.accept();

    r1.calculate();

    break;

case 2:

       circle c1;

    c1.accept();

    c1.calculate();

    break;

case 3:

   triangle t;

    t.accept();

    t.calculate();

    break;

case  4:

cout<<"Exit";

default:

cout<<"Enter valid choice"<<endl;

    break;

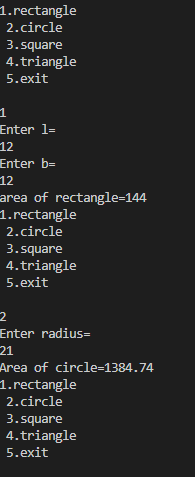
}

} while (ch!=4);

    return 0;

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Employee Inheritance:

Problem Statement: Build a system for managing employees. Create a base class Employee with attributes such as name, employee ID, and salary. Then, derive classes like Manager and Developer, each with its own attributes and methods. Implement a common method, like calculate\_salary(), in the base class.

#include <iostream>

#include <typeinfo>

using namespace std;

class employee

{

    int empid;

    string name;

    double salary;

public:

    employee()

    {

        cout << "Default constructor call";

        this->empid = 123;

        this->name = "Piyush";

        this->salary = 25000;

    }

    employee(int empid, double salary, string name)

    {

        this->name = name;

        this->empid = empid;

        this->salary = salary;

    }

    virtual void display()

    {

        cout << "Name : " << name << endl;

        cout << "empId : " << empid << endl;

        // cout<<"Salary : "<<salary<<endl;

    }

    double getsalary()

    {

        return salary;

    }

    void computeNetSalary()

    {

        cout << "Emp salary \n";

    }

};

class Manager : public employee

{

    int depid;

    double perfBonus = 12000;

public:

    Manager()

    {

        cout << "default mgr call\n";

    }

    Manager(string name, int depid, double salary, int empid) : employee(empid, salary, name)

    {

        this->depid = depid;

    }

    void computeNetSalary()

    {

        double s = employee::getsalary() + perfBonus;

        cout << "Salary with perBonus " << s << endl;

    }

    void display()

    {

        cout << "\*\*\*\*\*\*manager\*\*\*\*\*\*\*\*" << endl;

        employee::display();

        cout << "dep = " << depid << endl;

        computeNetSalary();

    }

};

class Developer : public employee

{

    int depid;

    double perfBonus = 12000;

public:

    Developer()

    {

        cout << "Bydefault" << endl;

    }

    Developer(int empid, double salary, string name, int depid) : employee(empid, salary, name)

    {

        this->depid = depid;

    }

    void computeNetSalary()

    {

        double s = employee::getsalary() + perfBonus;

        cout << "Salary with perBonus " << s << endl;

    }

    void display()

    {

        cout << "\*\*\*\*\*\*Developer\*\*\*\*\*\*\*\*" << endl;

        employee::display();

        cout << "Dep = " << depid << endl;

        computeNetSalary();

    }

};

int main()

{

    int empid = 1;

    employee \*pre[10];

    int ch, i = 0;

    string name;

    double salary;

    int depid;

    Developer w1;

    Manager m;

    do

    {

        cout << "1: manager \n2: Devloper\n3: Display\n4: Exit" << endl;

        cin >> ch;

        switch (ch)

        {

        case 1:

        {

            cout << "ENter the name : ";

            cin >> name;

            cout << "Enter the salary : ";

            cin >> salary;

            cout << "Enter the depid : ";

            cin >> depid;

            Manager \*m1 = new Manager(name, depid, salary, empid);

            pre[i] = m1;

            empid++;

            i++;

        }

        break;

        case 2:

        {

            cout << "Enter the name";

            cin >> name;

            cout << "Enter the salary : ";

            cin >> salary;

            cout << "Enter the depid : ";

            cin >> depid;

            Developer \*w1 = new Developer(empid, salary, name, depid);

            pre[i] = w1;

            i++;

            empid++;

        }

        break;

        case 3:

        {

            int j = 0;

            for (int j = 0; j < empid; j++)

            {

                pre[j]->display();

                cout << "\n";

                j++;

            }

        }

        break;

        case 4:

        {

            cout << "Exit\n";

        }

        break;

        default:

        {

            cout << "enter valid choice\n";

        }

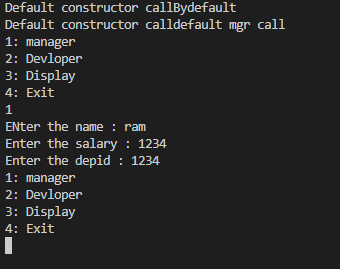
        break;

        }

    } while (ch != 4);

}

Output:



----------------------------------------------------------------------------------------------------------------------------------

Vehicle Inheritance:

Problem Statement: Develop a class hierarchy for vehicles. Start with a base class Vehicle and create derived classes like Car, Motorcycle, and Truck. Each derived class should have unique properties like the number of wheels and specific methods like start\_engine().

#include<iostream>

using namespace std;

class vehicles{

public:

 int wheels;

 virtual void accept(){

    cout<<"Enter no of wheels="<<endl;

    cin>>wheels;

 }

 virtual void startengine(){

    cout<<"Vehicle start"<<endl;

 }

};

class car: public vehicles{

public:

void accept(){

    cout<<"Enter car wheels="<<endl;

    cin>>wheels;

}

void startengine(){

    cout<<" wheels of car"<<wheels<<endl;

    cout<<"car start"<<endl;

}

};

class motorcycle: public vehicles{

public:

 void accept(){

    cout<<"Enter motorcycle wheel="<<endl;

    cin>>wheels;

 }

 void startengine(){

    cout<<"wheel of motorcylce"<<wheels<<endl;

    cout<<"motorcycle staet"<<endl;

 }

};

class truck: public vehicles

{

public:

void accept(){

    cout<<"Enter wheels="<<endl;

    cin>>wheels;

}

void startengine(){

    cout<<"Wgeel is="<<wheels<<endl;

    cout<<"truck is start"<<endl;

}

};

int main(){

motorcycle m1;

m1.accept();

m1.startengine();

car c1;

c1.accept();

c1.startengine();

truck t1;

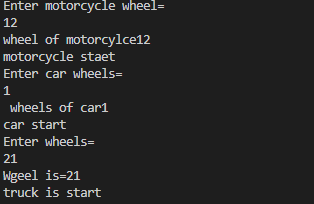
t1.accept();

t1.startengine();

    return 0;

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Bank Account Inheritance:

Problem Statement: Design a system for managing bank accounts. Create a base class BankAccount with attributes like account number and balance. Derive classes like SavingsAccount and CheckingAccount, each with specialized methods like withdraw() and calculate\_interest().

#include <iostream>

using namespace std;

class BankAccount

{

protected:

    int account\_number = 521512521;

    double balance = 10000;

public:

    int getbalance()

    {

        return balance;

    }

};

class SavingsAccount : public BankAccount

{

public:

    void withdraw(double amount)

    {

        this->balance -= amount;

        cout << amount << " withdraw....\n";

    }

    void deposite(double amount)

    {

        this->balance += amount;

        cout << amount << " deposite....\n";

    }

};

class CheckingAccount : public BankAccount

{

    int intrest;

public:

    void calculateIntrest(SavingsAccount act)

    {

        intrest = (act.getbalance() \* 0.02);

    }

    CheckingAccount(SavingsAccount act)

    {

        calculateIntrest(act);

        cout << "account checking.....\n";

        cout << "Your account balance is " << act.getbalance() << endl;

        cout << "interest = " << intrest << endl;

    }

};

int main()

{

    SavingsAccount act1;

    int ch, amount;

    cout << "1: withdrow\n2: deposite\n3: checkbalance\n4: exit" << endl;

    do

    {

        cout<<"Enter choice : ";

        cin>>ch;

        switch (ch)

        {

        case 1:

        {

            cout << "Enter amount for withdrow : ";

            cin >> amount;

            act1.withdraw(amount);

        }

        break;

        case 2:

        {

            cout << "Enter amount for deposite : ";

            cin >> amount;

            act1.deposite(amount);

        }

        break;

        case 3:

        {

            CheckingAccount a(act1);

        }

            break;

        case 4:

        {

            cout << "Have a good day !!" << endl;

        }

        break;

        default:

            cout << "Enter valid choice\n";

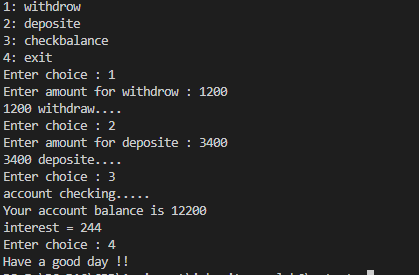
            break;

        }

    } while (ch != 4);

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Geometric Shapes with Polymorphism:

Problem Statement: Extend the shape hierarchy example by implementing polymorphism. Define a base class Shape with methods to calculate area and perimeter. Then, create derived classes like Circle, Rectangle, and Triangle, each with its own implementation of these methods.

#include <iostream>

using namespace std;

class shape

{

public:

    virtual void calculet\_area()=0;

    virtual void calculate\_perimeter()=0;

};

class Circle : public shape

{

    int r;

public:

    void calculet\_area()

    {

        cout << "enter redius : ";

        cin >> r;

        cout << "Area of circlue = " << (3.14 \* r \* r) << endl;

    }

    void calculate\_perimeter()

    {

        cout << "Perimeter of circle = " << (2 \* (3.14 \* r));

    }

};

class Rectangle : public shape

{

    int l, b;

public:

    void calculet\_area()

    {

        cout << "enter l and b : \n";

        cin >> l >> b;

        cout << "Area of rectangle = " << (l \* b) << endl;

    }

    void calculate\_perimeter()

    {

        cout << "Perimeter of rectangle = " << (2 \* (l + b));

    }

};

class Triangle : public shape

{

    int h, b;

    void calculet\_area()

    {

        cout << "enter base and hight : \n";

        cin >> b >> h;

        cout << "area of Tringle = " << ((h \* b) / 2) << endl;

    }

    void calculate\_perimeter()

    {

        int d;

        cout << "Enter side 1 : ";

        cin >> h;

        cout << "Enter side 2 : ";

        cin >> b;

        cout << "Enter side 3 : ";

        cin >> d;

        cout << "Perimeter of rectangle = " << (h + b + d);

    }

};

int main()

{

    Circle c;

    Rectangle r;

    Triangle t;

    shape \*sp[3];

    sp[0] = &c;

    sp[1] = &r;

    sp[2] = &t;

    for (int i = 0; i < 3; i++)

    {

        sp[i]->calculet\_area();

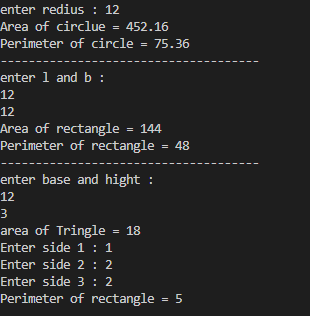
        sp[i]->calculate\_perimeter();

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

    }

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Person and Student Inheritance:

Problem Statement: Model a system for handling individuals and students within an educational institution. Create a base class Person with attributes like name and age. Derive a Student class with additional attributes like student ID and GPA, inheriting the common attributes from the Person class.

#include<iostream>

using namespace std;

class Person

{

    int age;

    string name;

    public:

    virtual void accept\_data()

    {

        cout<<"Enter name : ";

        cin>>name;

        cout<<"Enter age : ";

        cin>>age;

    }

    virtual void display()

    {

        cout<<"Name : "<<name<<endl;

        cout<<"age : "<<age<<endl;

    }

};

class Student:public Person

{

    int ID;

    int CGPA;

    public:

    void accept\_data()

    {

        cout<<"Enter Student details\n";

        Person::accept\_data();

        cout<<"Enter id : ";

        cin>>ID;

        cout<<"Enter CGPA : ";

        cin>>CGPA;

    }

    void display()

    {

        Person::display();

        cout<<"Id : "<<ID<<endl;

        cout<<"CGPA : "<<CGPA<<endl;

    }

};

class individuals:public Person

{

    public:

    void accept\_data(){

    cout<<"Enter person details\n";

    Person::accept\_data();

    }

    void display()

    {

        Person::display();

    }

};

int main()

{

    individuals man;

    man.accept\_data();

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

    man.display();

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

    Student st;

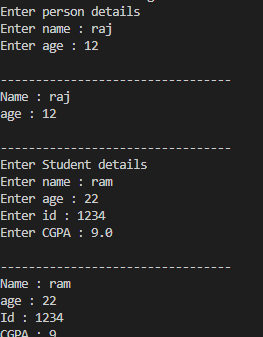
    st.accept\_data();

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

    st.display();

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Library Catalog with Books and Journals:

Problem Statement: Build a library catalog system. Create a base class LibraryItem with properties like title and author. Then, derive classes like Book and Journal, each with their unique properties. Implement methods to check out and return items in the derived classes.

#include<iostream>

using namespace std;

class libraryitem{

    private:  string title,author;

    public:

     libraryitem(){

        cout<<"default"<<endl;

        title="Science";

        author="Jayu";

     }

libraryitem(string title,string author){

    cout<<"parameter"<<endl;

    this->title=title;

    this->author=author;

}

void uniq(){

    cout<<"Book has very very knowledgable"<<endl;

}

virtual  void checkout(){

    cout<<"book are checked out"<<endl;

}

virtual void returne(){

    cout<<"book are returned"<<endl;

}

};

class book: public libraryitem{

private: string types;

public:

book(){

    cout<<"default"<<endl;

    types="cpp, java ,python";

}

book(string types){

this->types=types;

}

void uniq(){

    // libraryitem::uniq();

    cout<<"Programming language book"<<endl;

}

void checkout(){

    // libraryitem::checkout();

    cout<<"Programmable book checkout"<<endl;

}

void returne(){

    cout<<"returned"<<endl;

}

};

class journal:public libraryitem{

private: int jouid;

         double price;

public:

 journal(){

    cout<<"default"<<endl;

    jouid=101101;

    price=550;

 }

 journal(int jouid, double price){

this->jouid=jouid;

this->price=price;

 }

void uniq(){

    // libraryitem::uniq();

    cout<<"journals is helpful"<<endl;

}

void  checkout(){

    // libraryitem::checkout();

    cout<<"Checked out journals"<<endl;

}

void returne(){

    cout<<"Returned"<<endl;

}

};

int main(){

book b1;

b1.uniq();

b1.checkout();

b1.returne();

journal j1;

j1.uniq();

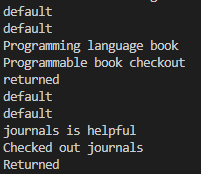
j1.checkout();

j1.returne();

    return 0;

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Shape Sorting with Interfaces:

Problem Statement: Implement a shape sorting program. Define a base class Shape with properties like area and perimeter. Create derived classes like Circle, Rectangle, and Triangle. Implement an interface Sortable with a method to compare shapes by area. Use this interface to sort a list of shapes.

#include<iostream>

using namespace std;

class shape{

private:

virtual void accept()=0;

virtual void area()=0;

virtual void perimeter()=0;

};

class circle: public shape{

int r;

public:

  void accept(){

    cout<<"Enter radius="<<endl;

    cin>>r;

  }

  void area(){

cout<<"area of circle="<<3.14\*r\*r<<endl;

  }

  void perimeter(){

    cout<<"perimeter of circle="<<2\*3.14\*r<<endl;

  }

};

class rectangle: public shape{

private:

int l,b;

public:

void accept(){

    cout<<"enter l,b="<<endl;

    cin>>l>>b;

}

void area(){

cout<<"area of rectangle="<<l\*b<<endl;

}

void perimeter(){

    cout<<"perimeter of rectangle"<<2\*l\*b<<endl;

}

};

class triangle: public shape{

private: int l, b,h;

public:

void accept(){

    cout<<"Enter length, and height="<<endl;

    cin>>b>>h;

}

void area(){

    cout<<"area of triangle="<<(b\*h/2)<<endl;

}

void perimeter(){

    cout<<"perimeter of triangle="<<l+b+h<<endl;

}

};

int main(){

int ch;

do

{

cout<<"Enter choice 1.circle \n  2.rectangle  \n 3.triangle  \n 4.exit \n";

cin>>ch;

switch (ch)

{

case 1:{

    circle c1;

    c1.accept();

    c1.area();

    c1.perimeter();}

    break;

case 2:{

 rectangle r1;

 r1.accept();

 r1.area();

 r1.perimeter();

 }

break;

case 3:{

 triangle t1;

 t1.accept();

 t1.area();

 t1.perimeter();}

break;

case 4:

 cout<<"Exit"<<endl;

break;

default:

cout<<"Enter valid choice"<<endl;

    break;

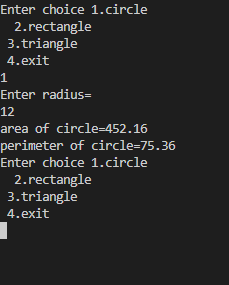
}

} while (ch !=4);

    return 0;

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Employee Payroll System with Abstract Classes:

Problem Statement: Design an employee payroll system. Create an abstract class Employee with attributes like name and employee ID. Derive concrete classes like HourlyEmployee and SalariedEmployee. Define abstract methods for calculating pay in the base class and implement them in the derived classes.

#include<iostream>

using namespace std;

class employee{

private: string name;

         int empid;

       protected:   double amount;

public:

  employee(){

cout<<"default"<<endl;

    name="Jayesh";

    empid=1234;

   amount=4000;

  }

  employee(string name, int empid,double amount){

    cout<<"parametr constr"<<endl;

this->name=name;

this->empid=empid;

this->amount=amount;

  }

  virtual void calculate(){

    cout<<"amount enter"<<endl;

    cin>>amount;

     cout<<"amount to be pay"<<amount<<endl;

  }

};

class hourly: public employee{

private: int hourlypay;

public:

hourly(){

    cout<<"hourly pay"<<endl;

}

void calculate(int hourly){

    amount=amount+hourly;

}

void dispaly(){

    cout<<"hourly"<<amount<<endl;

}

};

class salaryemp: public employee{

   public:

   void calculate(int hourly)

   {

    amount=amount+hourly;

    cout<<"salary"<<amount<<endl;

   }

void dispaly(){

    cout<<amount<<"salary= "<<endl;

cout<<"salary is gained"<<endl;

}

};

int main(){

 hourly h1;

 h1.calculate(1000);

 h1.dispaly();

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

 salaryemp s1;

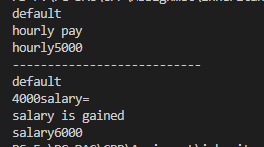
 s1.dispaly();

 s1.calculate(2000);

    return 0;

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Assignment 06

1 Solve this.

Fresh business scenario to apply inheritance , polymorphism to emp based organization scenario.

Create Emp based organization structure --- Emp , Mgr , Worker

1.1 Emp state--- id(int), name, deptId , basicSalary(double)

Accept all of above in constructor arguments

Methods ---1.2. compute net salary ---ret 0

(eg : public double computeNetSalary(){return 0;}

1.2 Mgr state ---id,name,basic,deptId , perfBonus

Add suitable constructor

Methods ----

1. compute net salary (formula: basic+perfBonus) -- override computeNetSalary

1.3 Worker state --id,name,basic,deptId,hoursWorked,hourlyRate

Methods :

1. compute net salary (formula: = basic+(hoursWorked\*hourlyRate) --override computeNetSalary

2. get hrlyRate of the worker -- add a new method to return hourly rate of a worker.(getter)

Create suitable array to store organization details.

Provide following options

1. Hire Manager

I/P : all manager details

2. Hire Worker

I/P : all worker details

3. Display information of all employees net salary (by invoking computeNetSal),

4. Exit

----------------------------------------------------

#include <iostream>

#include <typeinfo>

using namespace std;

class emp

{

    int empid;

    string name;

    double salary;

public:

    emp()

    {

        cout << "Default constructor call";

        this->empid = 123;

        this->name = "Piyush";

        this->salary = 25000;

    }

    emp(int empid, double salary, string name)

    {

        this->name = name;

        this->empid = empid;

        this->salary = salary;

    }

    virtual void display()

    {

        cout << "Name : " << name << endl;

        cout << "empId : " << empid << endl;

        // cout<<"Salary : "<<salary<<endl;

    }

    double getsalary()

    {

        return salary;

    }

    void computeNetSalary()

    {

        cout << "Emp salary \n";

    }

};

class mgr : public emp

{

    int depid;

    double perfBonus = 12000;

public:

    mgr()

    {

        cout << "default mgr call\n";

    }

    mgr(string name, int depid, double salary, int empid) : emp(empid, salary, name)

    {

        this->depid = depid;

    }

    void computeNetSalary()

    {

        double s = emp::getsalary() + perfBonus;

        cout << "Salary with perBonus " << s << endl;

    }

    void display()

    {

        cout<<"\*\*\*\*\*\*manager\*\*\*\*\*\*\*\*"<<endl;

        emp::display();

        cout << "dep = " << depid << endl;

        computeNetSalary();

    }

};

class Worker : public emp

{

    int id;

    string name;

    int deptid;

    int hoursWorked;

    int hourlyRate;

public:

    Worker()

    {

        cout << "Bydefault" << endl;

    }

    Worker(int empid, double salary, string name, int hoursWorked, int hourlyRate, int depid) : emp(empid, salary, name)

    {

        this->hoursWorked = hoursWorked;

        this->hourlyRate = hourlyRate;

        this->deptid = depid;

    }

    Worker(int empid,double salary,string name,int depid): emp(empid, salary, name){

        this->deptid = depid;

    }

    void computeNetSalary()

    {

        double s = emp::getsalary() + (hoursWorked \* hourlyRate);

        cout << "worker Salary = " << s << endl;

    }

    void display()

    {

        cout<<"\*\*\*\*\*\*Worker\*\*\*\*\*\*\*\*"<<endl;

        emp::display();

        cout << "Dep = " << deptid << endl;

        computeNetSalary();

    }

    int gethour()

    {

        return hourlyRate;

    }

};

int main()

{

    int empid = 1;

    emp \*pre[10];

    int ch, i = 0;

    string name;

    double salary;

    int depid;

    Worker w1;

    mgr m;

    do

    {

        cout << "1 for hire manager \n2 for hire Worker\n"

             << endl;

        cin >> ch;

        switch (ch)

        {

        case 1:

        {

            cout << "ENter the name : ";

            cin >> name;

            cout << "Enter the salary : ";

            cin >> salary;

            cout << "Enter the depid : ";

            cin >> depid;

            mgr \*m1 = new mgr(name, depid,salary, empid);

            pre[i] = m1;

            // pre[i] = &ptr;

            cout << "Manager is hired " << endl;

            empid++;

            i++;

        }

        break;

        case 2:

        {

            cout<<"Enter the name";

            cin>>name;

            cout << "Enter the salary : ";

            cin >> salary;

            cout << "Enter the depid : ";

            cin >> depid;

            Worker \*w1 = new Worker(empid,salary,name,depid);

            pre[i] = w1;

            cout<<"Worker is hired\n";

            i++;

            empid++;

        }

        break;

        case 3:

        {

            i = 0;

            for (int j = 0; j < empid; j++)

            {

                pre[i]->display();

                cout<<"\n";

                i++;

            }

        }

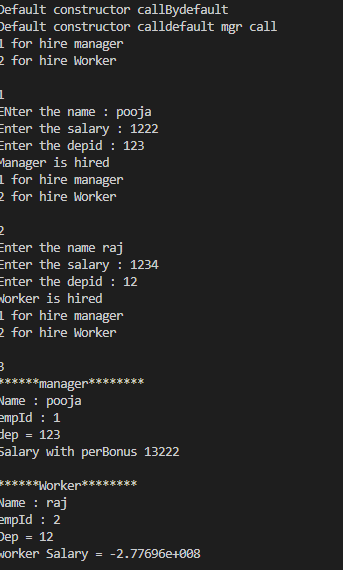
        break;

        }

    } while (ch != 4);

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

2:Create cpp application for bank account handling.

2.1. Create a class BankAccount -- acct no(int),customer name(string),balance(double)

Add constr. (2 constrs : first to accept all details )

2.2 Add Business logic methods

Methods

public void withdraw(double amt)

public void deposit(double amt)

2.3: Create object of account class and test withdraw and deposit methods.

**.h file**

#include<iostream>

using namespace std;

class account{

private: int actid;

         string name,city,email;

         double balance;

public:

     account(){

cout<<"Default account"<<endl;

   actid=1234;

   name="Jayesh";

   city="Pune";

   email="@qwe";

   balance=1000;

     }

     account(int actid,string name,string city,string email,double balance){

       cout<<"Parameter account"<<endl;

       this->actid=actid;

       this->name=name;

       this->city=city;

       this->email=email;

       this->balance=balance;

     }

void deposite(double amount){

    balance=balance+amount;

    cout<<"After deposite balance="<<balance<<endl;

}

void withdraw(double amount){

    balance=balance-amount;

    cout<<"after withdraw balance="<<balance<<endl;

}

~account(){

    cout<<"destrctor get called"<<endl;

}

void dispaly(){

    cout<<actid<<" "<<name<< " "<<city<<" "<<email<<" "<<balance<< ""<<endl;

}

void moneytransfer(double amount,account &receiveact){

this->balance=balance-amount;

receiveact.balance=receiveact.balance+amount;

cout<<"money transfer to"<<name<<"to"<<receiveact.name<<endl;

}

};

#include<iostream>

#include"bankk.h"

using namespace std;

int main()

{

    account ac1;

    ac1.deposite(1000);

    ac1.dispaly();

    account ac2(155,"jay","jal","ram@", 5000     );

    ac2.dispaly();

    ac2.deposite(1000);

ac1.moneytransfer(1000,ac2);

ac1.dispaly();

ac2.dispaly();

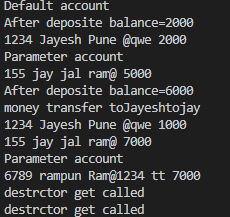
account \*ptr=new account(6789,"ram" "pun","Ram@1234","tt",7000);

ptr->dispaly();

    return 0;

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

3:Create a abstract class Shape with pure virtual method area;

Create Rectangle,Circle,Square class..inherit them from Shape class..Override area method.

Test these all classes by creating object of respective class.

#include <iostream>

using namespace std;

class shape

{

public:

    virtual void accept() = 0;

    virtual void calculte() = 0;

};

class rectangle : public shape

{

private:

    int l, b;

public:

    void accept()

    {

        cout << "Enter l=" << endl;

        cin >> l;

        cout << "Enter b=" << endl;

        cin >> b;

    }

    void calculte()

    {

        cout << "area of rectangle=" << (l \* b) << endl;

    }

};

class circle : public shape

{

private:

    int r;

public:

    void accept()

    {

        cout << "Enter radius=" << endl;

        cin >> r;

    }

    void calculte()

    {

        cout << "Area of circle=" << 3.14 \* r \* r << endl;

    }

};

class square : public shape

{

private:

    int side;

public:

    void accept()

    {

        cout << "Enter sides=" << endl;

    }

    void calculte()

    {

        cout << "area of square=" << side \* side << endl;

    }

};

class triangle : public shape

{

private:

    int b, h;

public:

    void accept()

    {

        cout << "Enter breadth and height=" << endl;

        cin >> b >> h;

    }

    void calculte()

    {

        cout << "area of triangle=" << (b \* h / 2) << endl;

    }

};

int main()

{

    int ch;

    do

    {

        cout << "1.rectangle \n 2.circle \n 3.square \n 4.triangle \n 5.exit \n"

             << endl;

        cin >> ch;

        switch (ch)

        {

        case 1:

        {

            rectangle r;

            r.accept();

            r.calculte();

        }

        break;

        case 2:

        {

            circle c1;

            c1.accept();

            c1.calculte();

        }

        break;

        case 3:

        {

            square sq;

            sq.accept();

            sq.calculte();

        }

        break;

        case 4:

        {

            triangle t1;

            t1.accept();

            t1.calculte();

        }

        break;

        case 5:

        {

        }

            cout << "exit" << endl;

            break;

        default:

            cout << "thank you....." << endl;

            break;

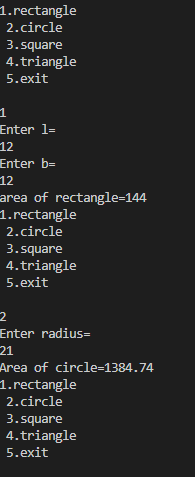
        }

    } while (ch != 4);

    return 0;

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Assignment 07

1:Create class for Handling Exception for Student

Create StudentException class;

Create Student class with data members (rollno,name,age)

at the time of student creation check age if age<18 then throw exception

#include<iostream>

using namespace std;

class Student

{

    int roll;

    string name;

    int age;

    public:

    Student(int r, string n, int a)

    {

        this->age = a;

        this->name = n;

        this->roll = r;

        cout<<"Student is created";

    }

};

int main()

{

    int roll,age;

    string name;

    try

    {

        cout<<"Enter the name"<<endl;

        cin>>name;

        cout<<"Enter the roll"<<endl;

        cin>>roll;

        cout<<"Enter the age"<<endl;

        cin>>age;

        if(age<18)

        {

            throw "age is below 18";

        }

        else

        {

            Student(roll,name,age);

        }

    }

    catch(const char \*str)

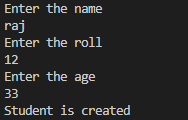
    {

       cout<<"Exeption : "<<str;

    }

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

2:Create Function Template and class template and use it

#include<iostream>

using namespace std;

template<class T> void sum(T a, T b)

{

    T t = a;

    a = b;

    b = t;

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

}

template<class C>class student

{

     public:

    C roll;

    C name;

   void accept(C r, C n)

    {

        this->roll = r;

        this->name = n;

    cout<<"roll = "<<roll<<" name = "<<name<<endl;

    }

};

template<class T>class MyGenClass{

    public:T val1;

    public:MyGenClass(T v){

        this->val1=v;

    }

    void print(){

        cout<<"Value1:"<<val1<<endl;

    }

};

int main()

{

    int a = 10, b = 20;

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

    sum(a,b);

    char a1 = 'A', b1 = 'C';

    cout<<"a = "<<a1<<" b = "<<b1<<endl;

    sum(a1,b1);

    student<string>m1;

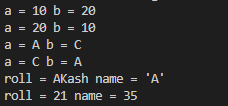
    m1.accept("AKash","'A'");

    student<int>m2;

    m2.accept(21,35);

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

3:Create abstract class Fruit with abstract function taste

Create derived classes for Fruit:Apple,Mango,Orange

Implement abstract method in derived class ,add specific methods like juice(),jam(),pulp() in respective derived class(as discussed in lect);

create taster program.create fruit basket and invoke method

Use:dynamic\_cast,typeid

#include<iostream>

#include<typeinfo>

using namespace std;

class fruit

{

    public:

    virtual void Test()

    {

        cout<<"no test";

    }

};

class apple:public fruit

{

    public:

    void Test()

    {

        cout<<"Apple is sweet\n";

    }

    void jam()

    {

        cout<<"Create Jam From Apple";

    }

};

class mango:public fruit

{

    public:

    void Test()

    {

        cout<<"Mango is sweet\n";

    }

    void pulp()

    {

        cout<<"Mango Pulp get Created...";

    }

};

class orange : public fruit

{

    public:

    void Test()

    {

        cout<<"Orange is sweet\n";

    }

    void juice()

    {

        cout<<"  Orange Taste Sweet  and sour";

    }

};

int main()

{

    fruit \*basket[5];

    orange o;

    apple a;

    mango m;

    basket[0] = &m;

    basket[1] = &o;

    basket[2] = &a;

    for (int i = 0; i < 5; i++)

    {

        cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

        basket[i]->Test();

        if((typeid(\*basket[i]))== typeid(mango))

        {

            mango \*m1 = dynamic\_cast<mango \*>(basket[i]);

            m1->pulp();

        }

        else if((typeid(\*basket[i]))== typeid(orange))

        {

             orange \*o = dynamic\_cast<orange \*>(basket[i]) ;

             o->juice();

        }

        else if((typeid(\*basket[i]))== (typeid(apple)))

        {

            apple \*a = dynamic\_cast<apple \*>(basket[i]);

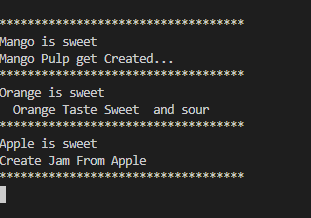
            a->jam();

        }

    }

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

Assignment 08

1 Write a program to store characters 'A' to 'Z' in the file .

Display the contents of file.

#include <iostream>

#include <fstream>

using namespace std;

int main()

{

    string st;

    ofstream file("file.txt");

    if (file.is\_open())

    {

        cout << "File is created\n";

    }

    file << "charactors from A to Z : ";

    for (char i = 'A'; i <= 'Z'; i++)

    {

        file<<i<<" ";

    }

    file.close();

    ifstream infile("file.txt");

    while (getline(infile, st))

    {

        cout << st;

    }

}

Output-



----------------------------------------------------------------------------------------------------------------------------------

2. Create class cEmployee with data members as empno,name and salary.

Accept values from user. Store it in file.

Display the contents from file.

Program should be able to store information of multiple employees.

#include <iostream>

#include <fstream>

using namespace std;

class employee

{

    int empno;

    char name[50];

    double salary;

public:

    void acceptData()

    {

        cout << "Name of employee : ";

        cin >> name;

        cout << "employee no : ";

        cin >> empno;

        cout << "employee salary : ";

        cin >> salary;

    }

    void display()

    {

        cout << "Name = " << name << endl;

        cout << "no = " << empno << endl;

        cout << "Salary = " << salary << endl;

    }

};

int main()

{

    employee e;

    e.acceptData();

    ofstream emp("emp.txt");

    if (emp.is\_open())

    {

        cout << "file created " << endl;

    }

    else

    {

        cout << "File not created " << endl;

    }

    emp.write((char \*)&e, sizeof(employee));

    cout << "data store succsefully !\n";

    emp.close();

    employee f;

    ifstream open("emp.txt");

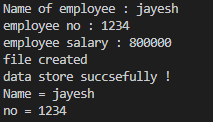
    open.read((char \*)&f, sizeof(employee));

    f.display();

    open.close();

}

Output-



----------------------------------------------------------------------------------------------------------------------------------

3:Write a program to copy the contents of one file and write it into another file and print it.

#include<iostream>

#include<fstream>

using namespace std;

int main()

{

    ofstream newfile("latest.txt");

    ifstream oldfile("oldfile.txt");

    string st;

    while (getline(oldfile,st))

    {

        newfile<<st<<endl;

    }

    cout<<"Contant stored !"<<endl;

    newfile.close();

    oldfile.close();

    ifstream check("latest.txt");

    while (getline(check,st))

    {

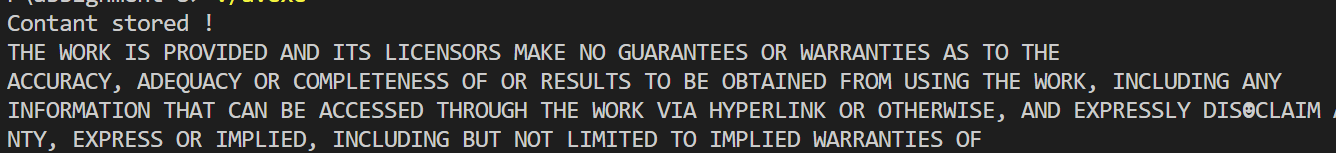
        cout<<st<<endl;

    }

    check.close();

}

Output=



----------------------------------------------------------------------------------------------------------------------------------

4:Create a C++ program that does the following:

Initializes an empty vector of integers.

Asks the user to enter a series of integers and adds them to the vector until the user enters a specific sentinel value (e.g., -1).

Prints the elements of the vector.

Calculates and prints the sum and average of the values in the vector.

#include <iostream>

#include <vector>

// #include <numeric>

using namespace std;

int main()

{

    vector<int> my;

    int ch, no;

    cout << "1: add 2: remove 3: display 4:exit" << endl;

    do

    {

        cout << "\nEnter choice : ";

        cin >> ch;

        switch (ch)

        {

        case 1:

        {

            cout << "Enter no for push : ";

            cin >> no;

            my.push\_back(no);

        }

        break;

        case 2:

        {

            my.pop\_back();

            cout<<"Popped !"<<endl;

        }

        break;

        case 3:

        {

            vector<int>::iterator i1 = my.begin();

            vector<int>::iterator j1 = my.end();

            while (i1 != j1)

            {

                cout << \*i1 << "  ";

                i1++;

            }

        }

        break;

        case 4:

                cout<<"Thank you !"<<endl;

                break;

        default:

        cout<<"Enter valid choice";

            break;

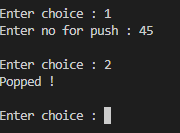
        }

    } while (ch != 4);

    return 0;

}

Output :



----------------------------------------------------------------------------------------------------------------------------------

5:Defines a Student class with attributes like name, age, and grade.

Initializes a vector of Student objects.

Allows the user to add new students to the vector with their name, age, and grade.

Prints the list of students.

#include <iostream>

#include <vector>

#include <numeric>

using namespace std;

class student

{

    int age;

    char grade;

    string name;

public:

    void accept()

    {

        cout << "Enter the name : ";

        cin >> name;

        cout << "enter the grade : ";

        cin >> grade;

        cout << "Enter age : ";

        cin >> age;

    }

    void display()

    {

        cout << "Name : " << name << endl;

        cout << "Age : " << age << endl;

        cout << "Grade : " << grade << endl;

    }

};

int main()

{

    vector<student> school;

    int ch;

    do

    {

        cout << "1: Add student\n2: display \n3:exit";

        cin>>ch;

        student st;

        switch (ch)

        {

        case 1:

        {

            st.accept();

            school.push\_back(st);

            cout << "Data stored !" << endl;

        }

        break;

        case 2:

        {

            int size = school.size();

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

            {

                school[i].display();

                cout << endl;

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

            }

        }

        break;

        case 3:

        {

            cout<<"Thank you !"<<endl;

        }

        default:

        {

            cout << "Enter valid choise" << endl;

        }

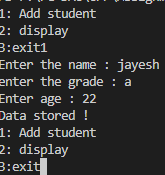
        break;

        }

    }while(ch!=3);

};

Output :



----------------------------------------------------------------------------------------------------------------------------------