Day 1

1:write program to test Hello World.

🡪

#include<iostream>

using namespace std;

int main(){

    cout<<"hello world";

}

Output:



2:Write a program to adddition of two numbers .

🡪

#include<iostream>

using namespace std;

int main(){

    int n1,n2,sum=0;

    cout<<"Enter two numbers:";

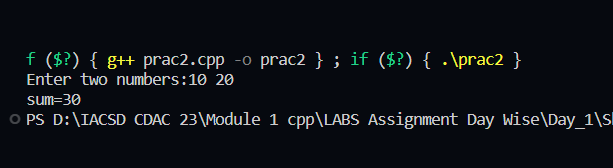
    cin>>n1>>n2;

    sum = n1+n2;

    cout<<"sum="<<sum;

}

Output:



3:Write a program to swap two numbers.

🡪

#include <iostream>

using namespace std;

void mySwap(int &x, int &y)

{

    int temp = x;

    x = y;

    y = temp;

}

int main()

{

    int a = 10, b = 20;

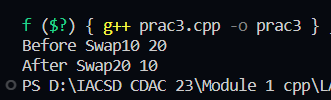
    cout<<"Before Swap"<<a<<" "<<b<<endl;

    mySwap(a,b);

    cout<<"After Swap"<<a<<" "<<b<<endl;

}

Output:



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

🡪

#include<iostream>

using namespace std;

int main(){

    int num;

    cout<<"Enter number";

    cin>>num;

    if(num%2==0){

        cout<<"Even";

    }

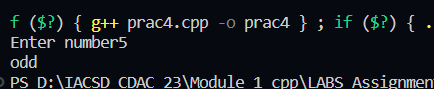
    else{

        cout<<"odd";

    }

}

Output:



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

🡪

#include <iostream>

using namespace std;

int main()

{

    int num;

    cout<<"enter number";

    cin>>num;

    if(num%5==0 && num%7==0)

    {

        cout<<"Number is Divisible by 5 and 7";

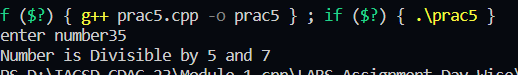
    }

    else

    cout<<"Not divisible";

}

Output:



6. Write a program, which accepts annual basic salary of an employee and calculates and displays the

Income 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>

 using namespace std;

 int main(){

    double basic,tax;

    cout<<"enter basic salary: ";

    cin>>basic;

    if(basic<150000){

        cout<<"no tax"<<endl;

    }

    else if(basic>=150000 && basic<300000){

        tax = basic\*20/100;

        cout<<"your tax: "<<tax<<endl;

    }

    else if(basic>300000){

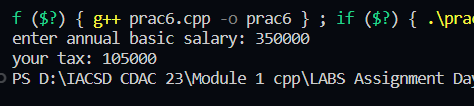
        tax = basic\*30/100;

        cout<<"your tax: "<<tax<<endl;

    }

 }

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>

using namespace std;

int main(){

    char ch;

    cout<<"Enter a lowercase alphabet:";

    cin>>ch;

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

        cout<<"vowel";

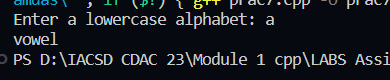
    }else{

        cout<<"consonant";

    }

}

Output:



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

🡪

#include<iostream>

using namespace std;

int main(){

    int a1,a2,a3,total=0;

    cout<<"enter 3 angles of triangle";

    cin>>a1>>a2>>a3;

    total = a1+a2+a3;

    if(total == 180){

        cout<<"Triangle is valid";

    }

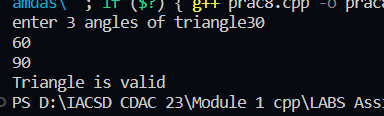
    else{

        cout<<"triangle is not valid";

    }

}

Output:



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

🡪

#include <iostream>

using namespace std;

void fact(int a)

{

    int fact = 1;

    if(a==0){

        fact = 1;

    }

    for(int i = 1; i <= a; i++)

    {

        fact \*= i;

    }

    cout<<fact;

}

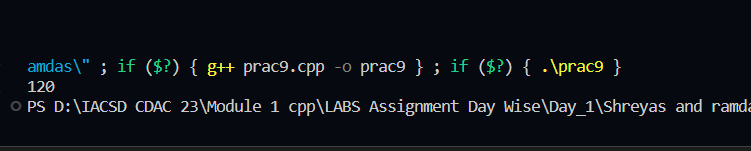
int main()

{

    fact(5);

}

Output:



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

🡪

#include<iostream>

using namespace std;

int power(int a, int b){

    int pow = 1;

    for(int i=b;i>0;i--){

         pow=pow\*a;

    }

//  while(b!=0){

//      pow \*=a;

//      b--;

//  }

//

    return pow;

}

int main(){

    int m,n;

    cout<<"Enter number: ";

    cin>>m;

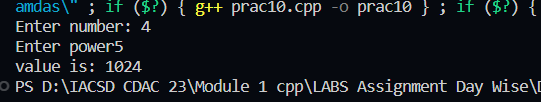
    cout<<"Enter power";

    cin>>n;

    cout<<"value is: "<<power(m,n)<<endl;

}

Output:



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

🡪

#include <iostream>

using namespace std;

void isPrime(int a)

{

    bool flag = false;

    for (int i = 2; i < a; i++)

    {

        if (a % i == 0)

        {

            flag = false;

            break;

        }

        else

        {

            flag = true;

        }

    }

    if (flag)

    {

        cout << "Prime";

    }

    else

        cout << "not prime";

}

int main()

{

    int num;

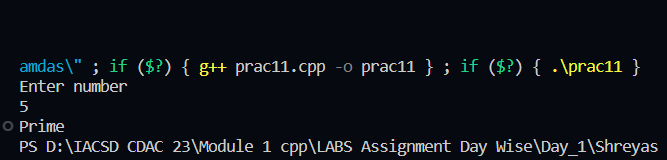
    cout << "Enter number" << endl;

    cin >> num;

    isPrime(num);

}

Output:



12:Sum of series :

1+2+3+….+n

🡪

#include<iostream>

using namespace std;

int sumofnos(int num){

    int sum=0;

    while(num != 0){

        sum +=num;

        num--;

    }

    return sum;

}

int main(){

    int num;

    cout<<"Enter number upto which you want to add: ";

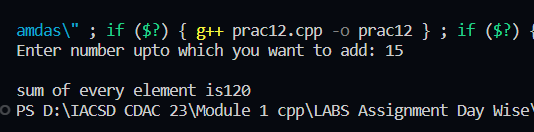
    cin>>num;

    cout<<"\n";

    cout<<"sum of every element is"<<sumofnos(num);

}

Output:



13:Check whether the number is palindrome or not?

🡪

#include <iostream>

using namespace std;

void isPalindrome(int a)

{

    int mod, res=0;

    int n = a;

    while(a>0){

        mod = a%10;

        res = res\*10 + mod;

        a = a/10;

    }

    if(res==n){

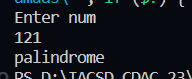
        cout<<"palindrome";

    }else{

        cout<<"not palindrome";

    }

Output:



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

🡪

#include<iostream>

using namespace std;

int main(){

    int n,sumeven = 0,sumodd = 0;

    cout<<"Enter n:";

    cin>>n;

    for(int i=1;i<=n;i++){

        if(i%2==0){

            sumeven+=i;

         }

         if(i%2!=0){

            sumodd+= i;

         }

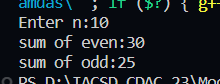
     }

     cout<<"sum of even:"<<sumeven;

     cout<<"sum of odd:"<<sumodd;

}

Output:



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

🡪

#include <iostream>

using namespace std;

void reverse(int a)

{

    int rev, mod;

    while (a > 0)

    {

        mod = a % 10;

        rev = rev \* 10 + mod;

        a /= 10;

    }

    cout << rev;

}

int main()

{

    int num;

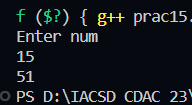
    cout << "Enter num" << endl;

    cin >> num;

    reverse(num);

}

Output:



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

🡪

#include <iostream>

using namespace std;

int main()

{

    int num, i, j;

    cout << "Enter number:";

    cin >> num;

    for (i = 1; i < num; i++)

    {

        for (j = 2; j <= i; j++)

        {

            if (i % j == 0)

            {

                break;

            }

        }

        if (i == j)

        {

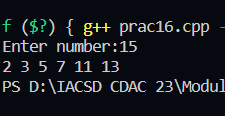
            cout << j << " ";

        }

    }

}

Output:



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

🡪

#include <iostream>

using namespace std;

void isAmstrong(int n)

{

    int mod, num, sum = 0, res = 0;

    num = n;

    while (n > 0)

    {

        mod = n % 10;

        sum = sum + (mod \* mod \* mod);

        n = n / 10;

    }

    if (sum == num)

    {

        cout << "number is amstrong";

    }

    else

    {

        cout << "num is not amstrong";

    }

}

int main()

{

    int num;

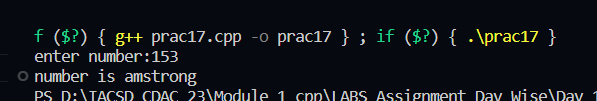
    cout << "enter number:";

    cin >> num;

    isAmstrong(num);

}

Output:



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

🡪

#include <iostream>

using namespace std;

int main()

{

    int n1, n2, n3;

    cout << "Enter the 3 numbers" << endl;

    cin >> n1 >> n2 >> n3;

    if (n1 < n2)

    {

        if (n3 < n2)

        {

            cout << "Max = " << n2;

        }

        else

        {

            cout << "Max = " << n3;

        }

    }

    else

    {

        if (n3 < n1)

        {

            cout << "Max = " << n1;

        }

        else

        {

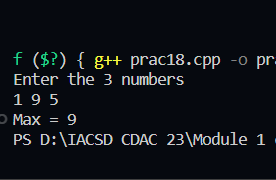
            cout << "Max = " << n3;

        }

    }

}

Output:



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

🡪

#include <iostream>

using namespace std;

int main()

{

    int choice, sum = 0, qty = 0;

    cout << "Select your pizza" << endl;

    cout << "1.margarita  200 \n2.farmhouse  350\n3.onion  150\n4.exit" << endl;

    do

    {

        cout << "enter choice";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "enter quantity:";

            cin >> qty;

            sum += qty \* 200;

            break;

        case 2:

            cout << "enter quantity:";

            cin >> qty;

            sum += qty \* 350;

            break;

        case 3:

            cout << "enter quantity:";

            cin >> qty;

            sum += qty \* 150;

            break;

        case 4:

            cout << "exited" << endl;

            break;

        default:

            cout << "invalid choice";

        }

    } while (choice != 4);

    cout << "Total:" << sum << endl;

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

}

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>

using namespace std;

int main()

{

    int num;

    cout << "enter single digit:";

    cin >> num;

    switch (num)

    {

    case 1:

        cout << "One";

        break;

    case 2:

        cout << "two";

        break;

    case 3:

        cout << "three";

        break;

    case 4:

        cout << "four";

        break;

    case 5:

        cout << "five";

        break;

    case 6:

        cout << "six";

        break;

    case 7:

        cout << "seven";

        break;

    case 8:

        cout << "eight";

        break;

    case 9:

        cout << "nine";

        break;

    default:

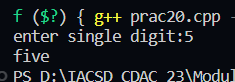
        cout << "invalid choice";

        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>

using namespace std;

int main(){

    int num1,num2;

    char ch;

    cout<<"enter two numbers:";

    cin>>num1>>num2;

    cout<<"\n 1.+ \n 2.- \n 3.\* \n 4./ \n 5.End(x)"<<endl;

    do{

            cout<<"Enter your choice:";

            cin>>ch;

            switch(ch){

                case '+':

                    cout<<"addition:"<<num1+num2<<endl;

                    break;

                case '-':

                    cout<<"substraction:"<<num1-num2<<endl;

                    break;

                case '\*':

                    cout<<"multi:"<<num1\*num2<<endl;

                    break;

                case '/':

                    cout<<"division:"<<num1/num2<<endl;

                    break;

                case 'x':

                    cout<<"exited"<<endl;

                    break;

                default:

                    cout<<"invalid choice"<<endl;

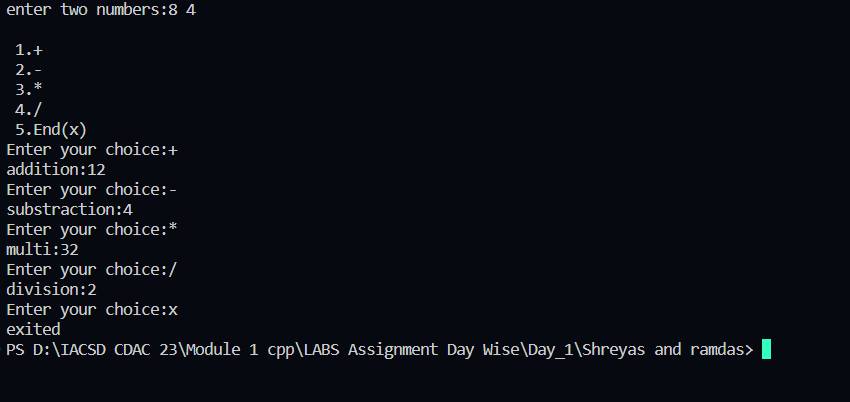
                    break;

            }

    }while(ch != 'x');

}

Output:



Day 2

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>

using namespace std;

int main()

{

    int num, sum = 0;

    cout << "Enter number:";

    cin >> num;

    while (num > 0)

    {

        sum += num;

        cin >> num;

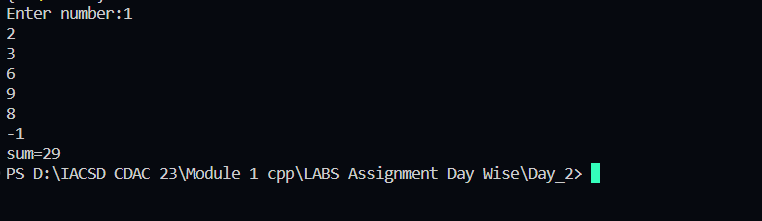
    }

    cout << "sum=" << sum;

    return 0;

}

Output:



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

🡪

#include <iostream>

using namespace std;

void power(int num, int power)

{

    int pow = 1;

    while (power > 0)

    {

        pow = pow \* num;

        power--;

    }

    cout << "answer=" << pow;

}

int main()

{

    int x, n;

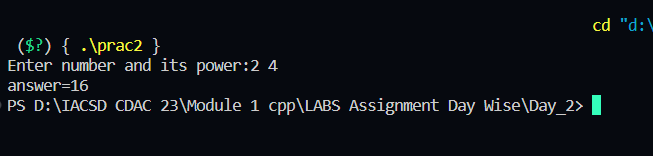
    cout << "Enter number and its power:";

    cin >> x >> n;

    power(x, n);

}

Output:



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

🡪

#include<iostream>

using namespace std;

int main(){

    char ch;

    int num,i=1;

    cout<<"Enter Number and character:";

    cin>>num>>ch;

    while(i<=num){

        ch = ch +1;

        if(ch == 'z'|| ch=='Z'){

            cout<<"Limit Reached";

            i=num;

        }

        else{

            cout<<ch<<"  ";

            i++;

        }

    }

}

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

For e.g. factorial of 5 = 5! = 5 \*4\*3\*2\*1 = 120

🡪

#include<iostream>

using namespace std;

int factorial(int n){

    int ans = 1;

    while(n>0){

        ans = ans\*n;

        n--;

    }

    return ans;

}

int main(){

    int num;

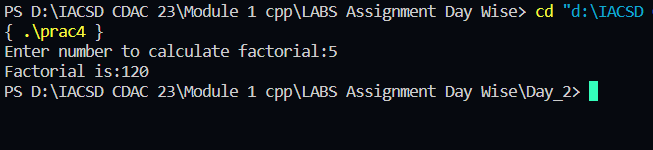
    cout<<"Enter number to calculate factorial:";

    cin>>num;

    cout<<"Factorial is:"<<factorial(num);

}

Output:



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

🡪

#include<iostream>

using namespace std;

int main(){

    int num;

    cout<<"Enter number to find factors of:";

    cin>>num;

    for(int i = 1;i<=num;i++){

        if(num % i == 0){

            cout<<i<<" ";

        }

    }

}

Output:

6. Accept two numbers and calculate GCD of them.

🡪

#include<iostream>

using namespace std;

int main(){

    int num1,num2,gcd;

    cout<<"Enter two numbers:";

    cin>>num1>>num2;

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

        if(num1 % i == 0 && num2 % i == 0){

            gcd = i;

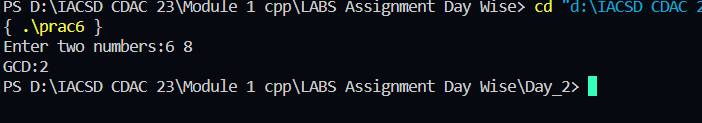
        }

    }

    cout<<"GCD:"<<gcd;

}

Output:



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

a) Compute area of circle

b) Compute area of rectangle

c) Compute area of triangle

d) 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>

using namespace std;

int main()

{

    char ch;

    cout<<"a) Compute area of circle \n b) Compute area of rectangle \nc) Compute area of triangle \nd) Exit";

    do{

    cout<<"\n Enter your choice:";

    cin>>ch;

    switch(ch){

        case 'a':

                int rad,area1;

                cout<<"Enter Radius:";

                cin>>rad;

                area1 = 3.14 \* rad\*rad;

                cout<<"area:"<<area1;

                break;

        case 'b':

                int l,b,area2;

                cout<<"Enter Length and Breadth";

                cin>>l>>b;

                area2 = l\*b;

                cout<<"area:"<<area2;

                break;

        case 'c':

                int h,base,area3;

                cout<<"Enter base and height";

                cin>>base>>h;

                area3 = 0.5\*b\*h;

                cout<<"area:"<<area3;

                break;

        case 'd':

                cout<<"Exited";

                break;

        default:

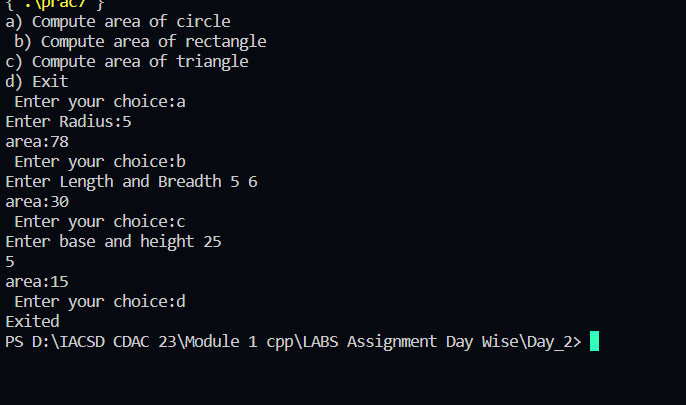
                cout<<"Invalid choice";

    }

    }while(ch != 'd');

}

Output:



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

🡪

#include <iostream>

using namespace std;

int main()

{

    int num, i, j;

    cout << "Enter number:";

    cin >> num;

    for (i = 1; i < num; i++)

    {

        for (j = 2; j <= i; j++)

        {

            if (i % j == 0)

            {

                break;

            }

        }

        if (i == j)

        {

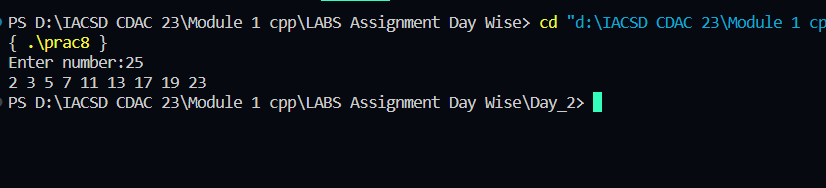
            cout << j << " ";

        }

    }

}

Output:



Day 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>

using namespace std;

class Student{

    int rollno,mark1,mark2,mark3;

    double total, percentage;

    char grade;

    public:

            void acceptInfo(){

                cout<<"Enter Roll no:";

                cin>>rollno;

                cout<<"\nEnter marks of 3 subjects out of 100:"<<endl;

                cin>>mark1>>mark2>>mark3;

            }

            void calculateMarks(){

                total = mark1+mark2+mark3;

                percentage = total/300 \* 100;

                if(percentage >=85){

                    grade = 'A';

                }

                else if(percentage >=75){

                    grade = 'B';

                }

                else if(percentage >=60){

                    grade = 'C';

                }

                else if(percentage >=40){

                    grade = 'D';

                }else{

                    grade = 'F';

                }

            }

            void display(){

                cout<<"Roll no:"<<rollno<<endl;

                cout<<"Marks:"<<mark1<<" "<<mark2<<" "<<mark3<<endl;

                cout<<"total:"<<total<<endl;

                cout<<"Percentage:"<<percentage<<endl;

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

            }

};

int main(){

    Student s1;

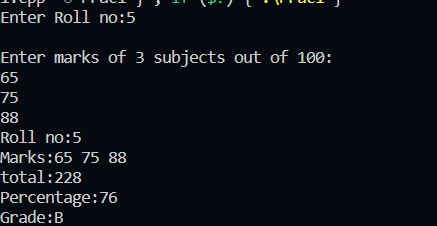
    s1.acceptInfo();

    s1.calculateMarks();

    s1.display();

}

Output:



2. 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>

using namespace std;

class Person

{

    string name;

    int age;

    string city;

public:

    Person(){

        name = "harshal";

        age =23;

        city="pune";

    }

    void setName(string name){

        this->name = name;

    }

    void setAge(int age){

        this->age = age;

    }

    void setCity(string city){

        this->city = city;

    }

    string getName(){

        return name;

    }

    int getAge(){

        return age;

    }

    string getCity(){

        return city;

    }

     void display(){

        cout<<"Name:"<<name<<endl;

        cout<<"age:"<<age<<endl;

        cout<<"city:"<<city<<endl;

    }

};

int main()

{

    string name;

    int age;

    string city;

    Person p1;

    cout<<"enter name:"<<endl;

    cin>>name;

    p1.setName(name);

    cout<<"Enter age:"<<endl;

    cin>>age;

    p1.setAge(age);

    cout<<"Enter city:"<<endl;

    cin>>city;

    p1.setCity(city);

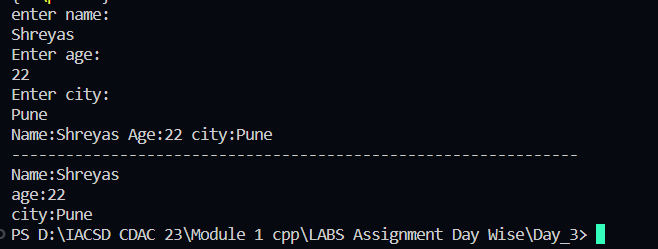
    cout<<"Name:"<<p1.getName()<<" Age:"<<p1.getAge()<<" city:"<<p1.getCity()<<endl;

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

    p1.display();

    return 0;

}



3. 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>

using namespace std;

class Date{

    private:

    int dd,mm,yyyy;

    public:

        void AcceptDate(){

            cout<<"Enter Date in dd/mm/yyyy format:";

            cin>>dd>>mm>>yyyy;

        }

        void DisplayDate(){

            cout<<dd<<"/"<<mm<<"/"<<yyyy;

        }

};

int main(){

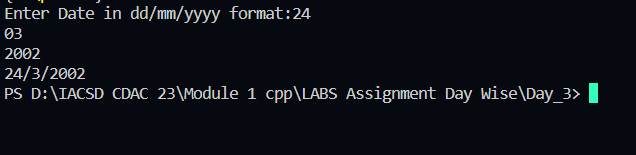
   Date d1;

   d1.AcceptDate();

   d1.DisplayDate();

}

Output:



4. 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>

using namespace std;

class Book{

    int id;

    double price;

    string bname;

    string author;

    public:

    void acceptData(){

        cout<<"Enter id:";

        cin>>id;

        cout<<"Enter name:";

        cin>>bname;

        cout<<"Enter author:";

        cin>>author;

        cout<<"Enter price:";

        cin>>price;

    }

    void display(){

        cout<<"book id:"<<id<<endl;

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

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

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

    }

};

int main(){

    Book b1;

    b1.acceptData();

    b1.display();

    Book b2;

    b2.acceptData();

    b2.display();

    Book b3;

    b3.acceptData();

    b3.display();

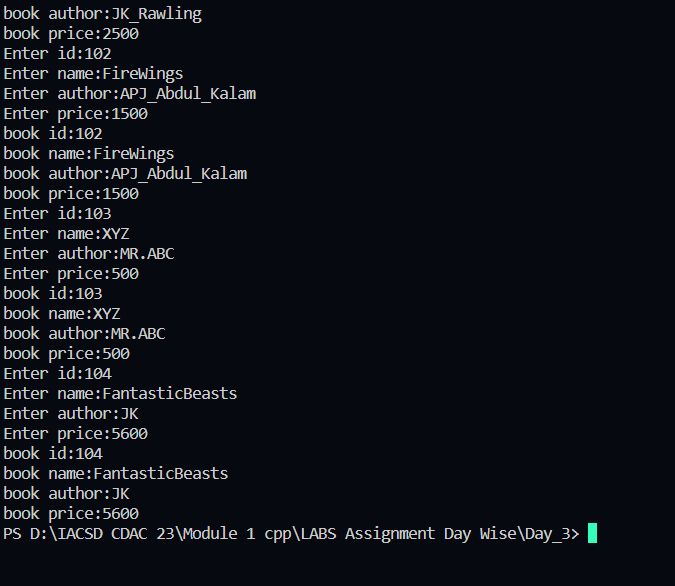
    Book b4;

    b4.acceptData();

    b4.display();

}

Output:



5. 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>

using namespace std;

class Point{

    double x,y;

    public:

            Point(){

                this->x = 5.2;

                this->y = 6.5;

            }

            Point(double x,double y){

                this->x = x;

                this->y = y;

            }

            void setX(double x){

                this->x=x;

            }

            double getX(){

                return x;

            }

            void setY(double y){

                this->y = y;

            }

            double getY(){

                return y;

            }

            void display(){

                cout<<"Point x:"<<this->x;

                cout<<"Point y:"<<this->y;

            }

};

int main(){

    double x,y;

    Point p1;

    p1.display();

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

    Point p2(3.7,4.5);

    p2.display();

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

    cout<<"Enter Point X and Y:";

    cin>>x>>y;

    Point p3(x,y);

    p3.display();

    cout<<"Change Point Y:";

    cin>>y;

    p3.setY(y);

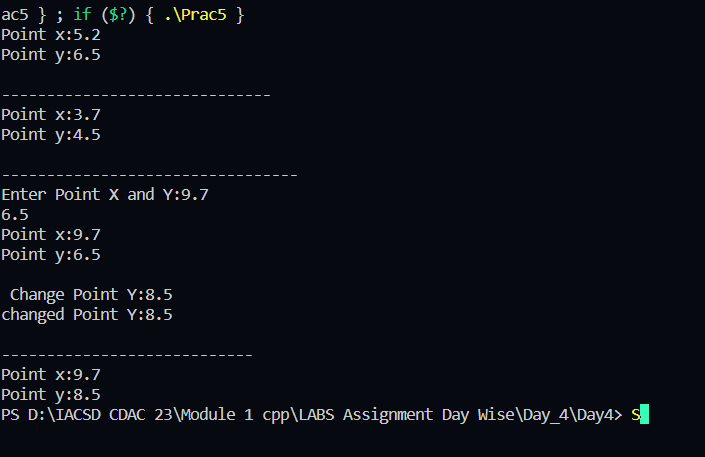
    cout<<"changed Point Y:"<<p3.getY()<<endl;

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

    p3.display();

}

Output:



6. 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.

🡪

//header file:

#include <iostream>

using namespace std;

class ComplexNumber

{

    int real, img;

    public:

        ComplexNumber()

        {

            real = 1;

            img = 2;

        }

        ComplexNumber(int real, int img)

        {

            this->real = real;

            this->img= img;

        }

        void setReal(int real)

        {

            this->real = real;

        }

        int getReal()

        {

            return real;

        }

        void setImg(int img)

        {

            this->img = img;

        }

        int getImg()

        {

            return img;

        }

        void display()

        {

            cout<<"Number is:"<<real<<"+"<<img<<"i"<<endl;

        }

};

//main code:

#include <iostream>

#include "MyComplex.h"

using namespace std;

int main()

  {

    ComplexNumber cn;

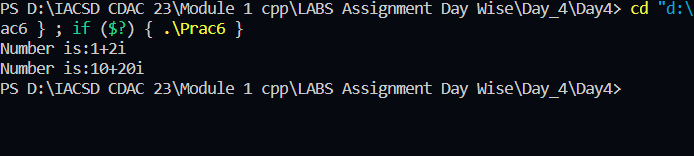
    cn.display();

    ComplexNumber cn2(10,20);

    cn2.display();

  }

Output:



Day 4

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>

using namespace std;

class Student{

    int rollno,mark1,mark2,mark3;

    double total, percentage;

    char grade;

    public:

            void acceptInfo(){

                cout<<"Enter Roll no:";

                cin>>rollno;

                cout<<"\nEnter marks of 3 subjects out of 100:"<<endl;

                cin>>mark1>>mark2>>mark3;

            }

            void calculateMarks(){

                total = mark1+mark2+mark3;

                percentage = total/300 \* 100;

                if(percentage >=85){

                    grade = 'A';

                }

                else if(percentage >=75){

                    grade = 'B';

                }

                else if(percentage >=60){

                    grade = 'C';

                }

                else if(percentage >=40){

                    grade = 'D';

                }else{

                    grade = 'F';

                }

            }

            void display(){

                cout<<"Roll no:"<<rollno<<endl;

                cout<<"Marks:"<<mark1<<" "<<mark2<<" "<<mark3<<endl;

                cout<<"total:"<<total<<endl;

                cout<<"Percentage:"<<percentage<<endl;

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

            }

};

int main(){

    Student s1;

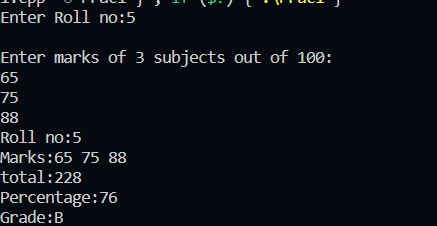
    s1.acceptInfo();

    s1.calculateMarks();

    s1.display();

}

Output:



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

using namespace std;

class Person{

    string name, city;

    int age;

    public:

    Person()

    {

        age = 24;

        name = "Sandesh";

        city = "Pune";

    }

    Person(string name, string city, int age)

    {

        this->name = name;

        this->city = city;

        this->age = age;

    }

    void setName(string name)

    {

        this->name = name;

    }

    string getName()

    {

        return name;

    }

    void setCity(string city)

    {

        this->city = city;

    }

    string getCity()

    {

        return city;

    }

    void setAge(int age)

    {

        this->age= age;

    }

    int getAge()

    {

        return age;

    }

    void display()

    {

        cout<<"Name: "<<name<<endl;

        cout<<"City: "<<city<<endl;

        cout<<"Age: "<<age<<endl;

    }

};

int main()

{

    Person p1;

    p1.display();

    Person p2("Shreyas", "Pune", 21);

    p2.display();

    Person p3;

    int age;

    string name, city;

    cout<<"Enter name, age, city:"<<endl;

    cin>>name>>age>>city;

    p3.setName(name);

    p3.setAge(age);

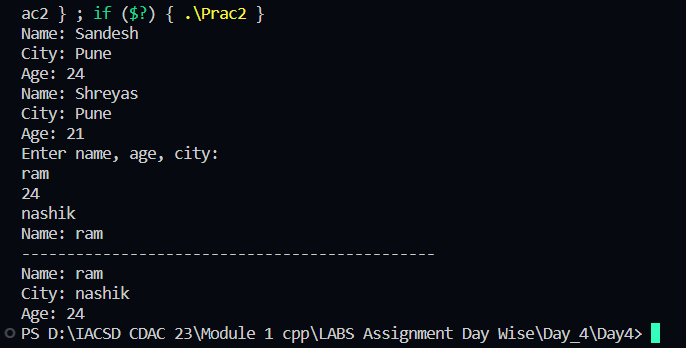
    p3.setCity(city);

    cout<<"Name: "<<p3.getName()<<endl;

    p3.display();

}

Output:



3. 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>

using namespace std;

class Date{

    int dd,mm,yy;

    public:

            Date(){

                this->dd = 1;

                this->mm = 6;

                this->yy = 1998;

            }

            Date(int dd,int mm,int yy){

                this->dd = dd;

                this->mm = mm;

                this->yy = yy;

            }

            void setDD(int dd){

                this->dd = dd;

            }

            int getDD(){

                return dd;

            }

            void setMM(int mm){

                this->mm = mm;

            }

            int getMM(){

                return mm;

            }

            void setYY(int yy){

                this->yy = yy;

            }

            int getYY(){

                return yy;

            }

            void display(){

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

            }

};

int main(){

    Date d1;

    d1.display();

    Date d2(24,3,2002);

    d2.display();

    int dd;

    cout<<"Change Date:";

    cin>>dd;

    d2.setDD(dd);

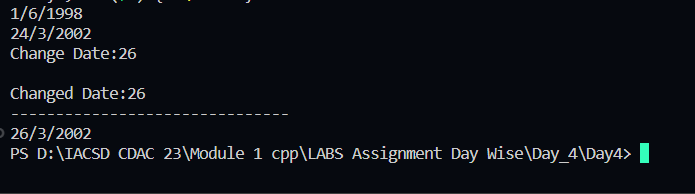
    cout<<"\nChanged Date:"<<d2.getDD();

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

    d2.display();

}

Output:



4. 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>

using namespace std;

class Book

{

    string bName, author;

    double price;

    int id;

    public:

        Book()

        {

            bName = "ABC";

            author = "mr xyz";

            price = 500.00;

            id = 1;

        }

        Book(int id,string bName, string author, double price)

        {

            this->id = id;

            this->bName = bName;

            this->author = author;

            this->price = price;

        }

        void setID(int id)

        {

            this->id = id;

        }

        int getID()

        {

            return id;

        }

        void setBookName(string bName)

        {

            this->bName = bName;

        }

        string getBookName()

        {

            return bName;

        }

        void setAuthor(string author)

        {

            this->author = author;

        }

        string getAuthor()

        {

            return author;

        }

        void setPrice(int price)

        {

            this->price = price;

        }

        int getPrice()

        {

            return price;

        }

        void display()

        {

            cout<<"Id: "<<id<<endl;

            cout<<"Name: "<<bName<<endl;

            cout<<"Author: "<<author<<endl;

            cout<<"Price: "<<price<<endl;

        }

};

int main()

{

    Book b1;

    b1.display();

    Book b2(10, "Fire", "MR APJ", 500);

    b2.display();

    string bName, author;

    double price;

    int id;

    cout<<"Enter book id, name, author, price"<<endl;

    cin>>id>>bName>>author>>price;

    Book b4(id, bName, author, price);

    b4.display();

    cout<<"Change Author"<<endl;

    cin>>author;

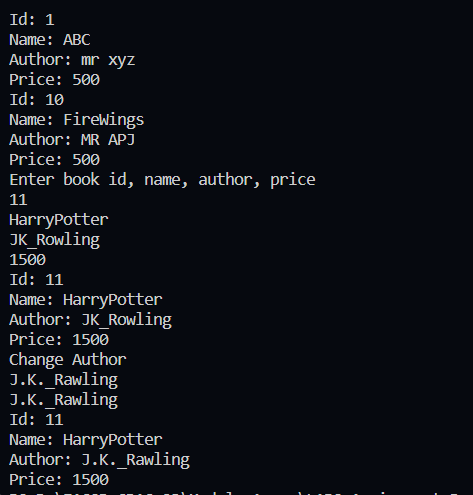
    b4.setAuthor(author);

    cout<<b4.getAuthor()<<"\n";

    b4.display();

}

Output:



5. 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>

using namespace std;

class Point{

    double x,y;

    public:

            Point(){

                this->x = 5.2;

                this->y = 6.5;

            }

            Point(double x,double y){

                this->x = x;

                this->y = y;

            }

            void setX(double x){

                this->x=x;

            }

            double getX(){

                return x;

            }

            void setY(double y){

                this->y = y;

            }

            double getY(){

                return y;

            }

            void display(){

                cout<<"Point x:"<<this->x;

                cout<<"Point y:"<<this->y;

            }

};

int main(){

    double x,y;

    Point p1;

    p1.display();

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

    Point p2(3.7,4.5);

    p2.display();

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

    cout<<"Enter Point X and Y:";

    cin>>x>>y;

    Point p3(x,y);

    p3.display();

    cout<<"Change Point Y:";

    cin>>y;

    p3.setY(y);

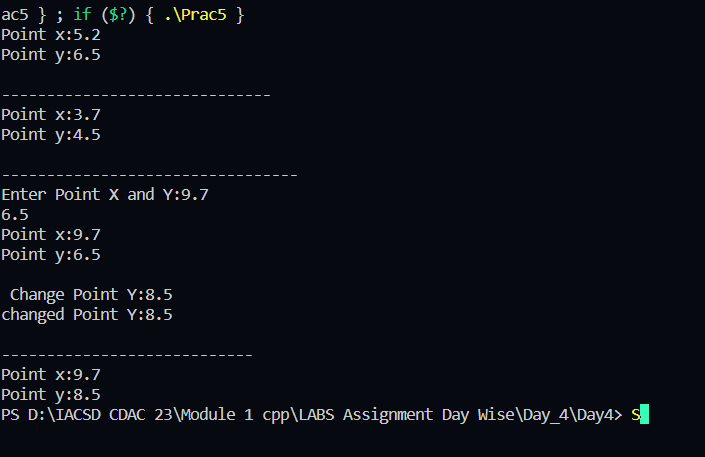
    cout<<"changed Point Y:"<<p3.getY()<<endl;

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

    p3.display();

}

Output:



6. 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.

🡪

//header file:

#include <iostream>

using namespace std;

class ComplexNumber

{

    int real, img;

    public:

        ComplexNumber()

        {

            real = 1;

            img = 2;

        }

        ComplexNumber(int real, int img)

        {

            this->real = real;

            this->img= img;

        }

        void setReal(int real)

        {

            this->real = real;

        }

        int getReal()

        {

            return real;

        }

        void setImg(int img)

        {

            this->img = img;

        }

        int getImg()

        {

            return img;

        }

        void display()

        {

            cout<<"Number is:"<<real<<"+"<<img<<"i"<<endl;

        }

};

//main code:

#include <iostream>

#include "MyComplex.h"

using namespace std;

int main()

  {

    ComplexNumber cn;

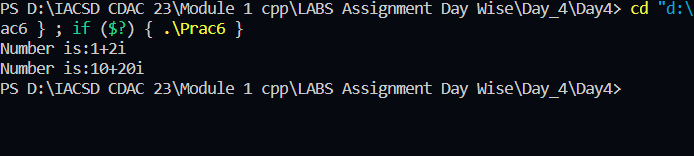
    cn.display();

    ComplexNumber cn2(10,20);

    cn2.display();

  }

Output:



Day 5

1:Create Date class with members day,month ,year.

Write no argument and parameterised constructor .Create two object s and initialize them using no argument and parameterised constructor

respectively.Print date using display function.

🡪

#include <iostream>

using namespace std;

class Date

{

    private:

        int dd, mm, yy;

    public:

        Date()

        {

            this->dd = 20;

            this->mm = 10;

            this->yy = 2020;

        }

        Date(int dd, int mm, int yy)

        {

            this->dd = dd;

            this->mm = mm;

            this->yy = yy;

        }

        void display()

        {

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

        }

};

int main()

{

    Date d1;

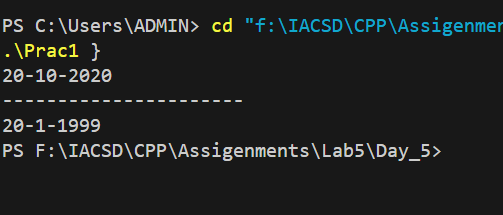
    d1.display();

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

    Date d2(20, 01, 1999);

    d2.display();

}



2:Create Employee class with members id(int),name(string),dob(Date).Use above created Date class.

Write default and parameterised constructor in Employee Class.Write accept() function to accept information and display() to display emp information.

🡪

#include <iostream>

using namespace std;

class Date

{

    private:

        int dd, mm, yy;

    public:

        Date()

        {

            this->dd = 20;

            this->mm = 10;

            this->yy = 2020;

        }

        Date(int dd, int mm, int yy)

        {

            this->dd = dd;

            this->mm = mm;

            this->yy = yy;

        }

        void display()

        {

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

        }

};

class Employee{

    private:

        int id;

        string name;

        Date dob;

    public:

        Employee():dob(){

            this->id = 101;

            this->name = "atharva";

            this->dob = Date(20, 10, 2020);

        }

        Employee(int id,string name, Date dob){

            this->id = id;

            this->name = name;

            this->dob = dob;

        }

        void accept(int id,string name, Date dob)

        {

            this->id = id;

            this->name = name;

            this->dob = dob;

        }

        void display()

        {

            cout<<id<<"  "<<name<<"  "<<endl;

            dob.display();

        }

};

int main()

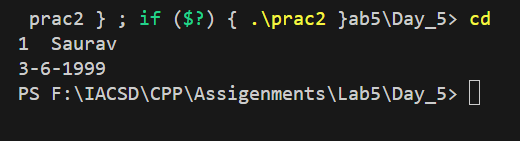
{

    Date d1(3, 6, 1999);

    Employee e1(1, "Saurav", d1);

    e1.display();

}



3:Consider that payroll software needs to be developed for computerization of

operations of an ABC organization. The organization has employees.

3.1. Construct a class Employee with following members using private access

specifies:

Employee Id integer

Employee Name string

Basic Salary double

HRA double

Medical double=1000

PF double

PT double

Net Salary double

Gross Salary double

Please use following expressions for calculations://Note:Don't accept HRA,PF PT from user

\* HRA = 50% of Basic Salary

\* PF = 12% of Basic Salary

\* PT = Rs. 200

3.2. Write methods to display the details of an employee and calculate the gross

and net salary.

\* Goss Salary = Basic Salary + HRA + Medical

\* Net Salary = Gross Salary – (PT + PF)

Create Object of employee class and assign values and display Details.

🡪

#include <iostream>

using namespace std;

class Employee

{

    private:

        int id;

        string name;

        double salary, hra, medical, pf, pt, netSalary, basicSalary,grossSalary;

    public:

        Employee(int id, string name, double basicSalary)

        {

            this->id = id;

            this->name = name;

            this->basicSalary = basicSalary;

            this->netSalary = netSalary;

        }

        void calSalary()

        {

            hra = (50 \* basicSalary / 100);

            pf = (12 \* basicSalary / 100);

            pt = 200;

            medical = 1000;

            grossSalary = basicSalary + hra + medical;

            netSalary = grossSalary - (pt + pf);

        }

        void display()

        {

            cout<<id<<"\t"<<name<<"\t"<<basicSalary<<endl;

            cout<<"pf: "<<pf<<"\t"<<"pt: "<<pt<<endl;

            cout<<"Net Salary: "<<netSalary<<endl;

            cout<<"Gross Salary: "<<grossSalary<<endl;

        }

};

int main()

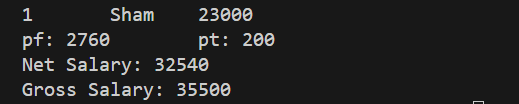
{

    Employee e1(1, "Sham", 23000);

    e1.calSalary();

    e1.display();

}



Day 6

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>

using namespace std;

class Employee{

    int id;

    string name;

    int deptId;

    protected:

    double basicSalary;

    public:

        Employee(int id,string name,int deptId,double basicSalary){

            this->id = id;

    this->name = name;

    this->deptId = deptId;

    this->basicSalary = basicSalary;

        }

        virtual double computeNetSalary(){

            return 0;

        }

        virtual void display(){

            cout<<"id:"<<id<<endl;

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

            cout<<"deptId:"<<deptId<<endl;

            cout<<"salary:"<<basicSalary<<endl;

        }

};

class Manager:public Employee{

    private:

        double perfBonus;

    public:

        Manager(int id,string name,int deptId,double basicSalary,double perfBonus):Employee(id,name,deptId,basicSalary){

                this->perfBonus = perfBonus;

            }

        double computeNetSalary(){

            basicSalary=basicSalary + perfBonus;

            return basicSalary;

        }

        void display(){

            cout<<"Net Salary:"<<basicSalary;

        }

};

class Worker:public Employee

{

    private:

            int hoursWorked, hourlyRate;

    public:

            Worker(int id,string name,int deptId,double basicSalary, int hourlyRate, int hoursWorked):Employee(id,name,deptId,basicSalary)

            {

                this->hourlyRate = hourlyRate;

                this->hoursWorked = hoursWorked;

            }

            double computeNetSalary()

            {

                basicSalary = basicSalary + (hoursWorked\*hourlyRate);

                return basicSalary;

            }

            int getHrlyRate()

            {

                return hourlyRate;

            }

};

#include<iostream>

#include "Employee.h"

using namespace std;

int main()

{

int id,deptId,hrlyRate,hrsWorked;

string name;

double basic,perfBonus;

int choice;

cout<<"1.Hire Manager\n2.Hire Worker\n3.Display information of all employees net salary \n4.exit";

Employee \*emp[2];

do{

    cout<<"Enter choice:";

    cin>>choice;

    switch(choice){

        case 1:

            {cout<<"-----------Hire Manager--------"<<endl;

            cout<<"Enter id, name,deptId,basic,perfBonus: ";

            cin>>id>>name>>deptId>>basic>>perfBonus;

            emp[0] = new Manager(id, name,deptId, basic, perfBonus);

            cout<<"Manager Registered";

            break;}

        case 2:

            {cout<<"-----------Hire Worker--------"<<endl;

            cout<<"Enter id, name,deptId,basic,Hourly Rate and Hours Worked: ";

            cin>>id>>name>>deptId>>basic>>hrlyRate>>hrsWorked;

            emp[1]= new Worker(id, name,deptId, basic, hrlyRate,hrsWorked);

            cout<<"Worker Registered";

            break;}

        case 3:

            {for(int i = 0;i<2;i++){

            cout<<"Salary:"<<emp[i]->computeNetSalary()<<endl;

            }

            break;}

        case 4:

            {cout<<"Exited";

            break;}

        default:

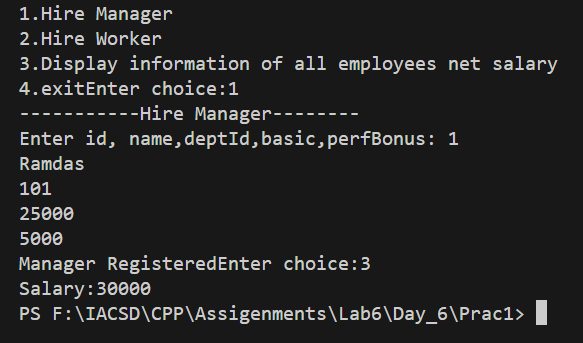
            {cout<<"Invalid Choice";

            break;}

    }

}while(choice!=4);

}



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.

🡪

#include<iostream>

#include "BankAccount.h"

using namespace std;

BankAccount::BankAccount(){

    this->acct\_no=101;

    this->name = "ABC";

    this->balance = 1000;

}

BankAccount::BankAccount(int acct\_no,string name,double balance){

    this->acct\_no = acct\_no;

    this->name = name;

    this->balance = balance;

}

void BankAccount::withdraw(double amount){

    balance = balance-amount;

    cout<<"Withdraw successful bank balance is:"<<balance<<endl;

}

void BankAccount::deposite(double amount){

    balance = balance+amount;

    cout<<"deposite successful bank balance is:"<<balance<<endl;

}

void BankAccount::display(){

    cout<<acct\_no<<"  "<<name<<"  "<<balance<<endl;

}

int main(){

    BankAccount act1;

    act1.display();

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

    act1.deposite(50000);

    act1.display();

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

    act1.withdraw(10000);

    act1.display();

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

    // BankAccount act2(102,"sham",15000);

    // act2.display();

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

    // act2.deposite(15000);

    // act2.display();

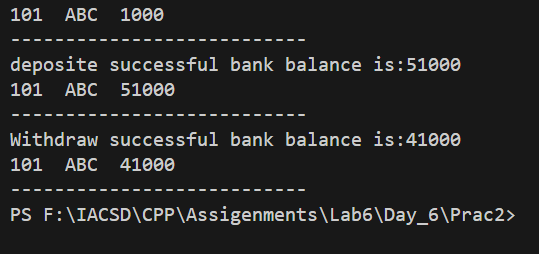
        // cout<<"---------------------------"<<endl;

    // act2.deposite(10000);

    // act2.display();

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

}



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 area() = 0;

};

class Rectangle: public Shape

{

    public :

    void area()

    {

        cout<<"Inside Rectangle class"<<endl;

    }

};

class Circle: public Shape

{

    public :

    void area()

    {

        cout<<"Inside Circle class"<<endl;

    }

};

class Square: public Shape

{

    public :

    void area()

    {

        cout<<"Inside Square class"<<endl;

    }

};

int main()

{

//  Shape s1;

//  s1.area();

Rectangle r1;

r1.area();

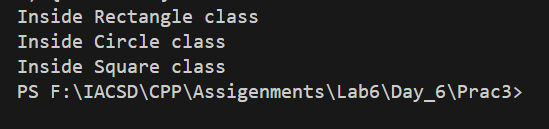
Circle c1;

c1.area();

Square s1;

s1.area();

}



Day 7

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 StudentException: public exception

{

    public:

        string msg;

        StudentException (string msg)

        {

            this->msg = msg;

        }

        void display()

        {

            cout<<"Age is less than 18 "<<endl;

        }

};

class Student

{

    private:

        int rollno, age;

        string name;

    public:

        Student(int rollno, int age, string name)

        {

            this->rollno = rollno;

            this->age = age;

            this->name = name;

        }

        void display()

        {

            StudentException ex("");

            if(age < 18)

            {

                throw ex;

            }

            cout<<rollno<<"  "<<name<<"  "<<age<<endl;

        }

};

int main()

{

    Student s1(1, 24, "Ayaush");

    try

    {

        s1.display();

    }

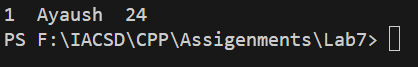
    catch(StudentException ex)

    {

        ex.display();

    }

}



2:Create Function Template and class template and use it

🡪

#include <iostream>

using namespace std;

template <class T> class Hello

{

    private:

        T a;

    public:

        Hello(T a)

        {

            this->a = a;

        }

        void display()

        {

            cout<<"A = "<<a<<endl;

        }

};

template <class A> void swap1(A &a, A &b)

{

    A temp;

    temp = a;

    a = b;

    b = temp;

    cout<<"A: "<<a<<"\t"<<"B: "<<b<<endl;

}

int main()

{

    int x = 10, y = 20;

    swap1(x, y);

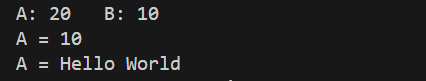
    Hello <int> h1(10);

    h1.display();

    Hello <string> h2("Hello World");

    h2.display();

}



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

{

    private:

        string name;

        string color;

        int weight;

    public:

        virtual void taste() = 0;

};

class Apple:public Fruit

{

    public:

        void taste()

        {

            cout<<"Apple is sweet"<<endl;

        }

        void jam()

        {

            cout<<"Creating jam"<<endl;

        }

};

class Mango: public Fruit

{

    public:

        void taste()

        {

            cout<<"Mango is sweet"<<endl;

        }

        void pulp()

        {

            cout<<"Creating pulp"<<endl;

        }

};

class Orange: public Fruit

{

    public:

        void taste()

        {

            cout<<"Orange is sour"<<endl;

        }

        void juice()

        {

            cout<<"Creating juice"<<endl;

        }

};

int main()

{

    Apple a1;

    a1.taste();

    a1.jam();

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

    Mango m1;

    m1.taste();

    m1.pulp();

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

    Orange o1;

    o1.taste();

    o1.juice();

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

    Fruit \*basket[3];

    basket[0] = &a1;

    basket[1] = &m1;

    basket[2] = &o1;

    for(int i = 0; i < 3; i++)

    {

        // basket[i]->taste();

        if(typeid(Apple) == typeid(basket[i]))

        {

            basket[i]->taste();

            (dynamic\_cast <Apple\*> (basket[i]))->jam();

        }

        else if(typeid(Mango) == typeid(basket[i]))

        {

            basket[i]->taste();

            (dynamic\_cast <Mango\*> (basket[i]))->pulp();

        }

        else

        {

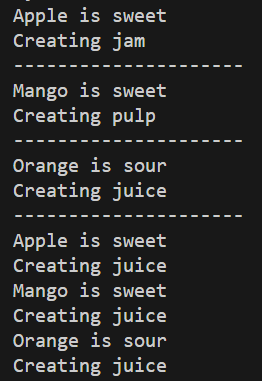
            basket[i]->taste();

            (dynamic\_cast <Orange\*> (basket[i]))->juice();

        }

    }

}



Day 8

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()

{

    //  fstream MyChars;

    string alphabet;

    ofstream MyChars("MyChar.txt");

    if (MyChars.is\_open())

    {

        cout << "File Created" << endl;

    }

    else

    {

        cout << "Creation failed" << endl;

    }

    for (int i = 65; i < 91; i++)

    {

        //      cout<<(char)i<<"  "<<endl;

        alphabet = (char)i;

        MyChars << alphabet << endl;

    }

    MyChars.close();

    ifstream inputFile("MyChar.txt");

    if (inputFile.is\_open())

    {

        cout << "file open" << endl;

        while (getline(inputFile, alphabet))

        {

            // inputFile>>line;

            cout << alphabet << endl;

            // cout.put(alphabet);

        }

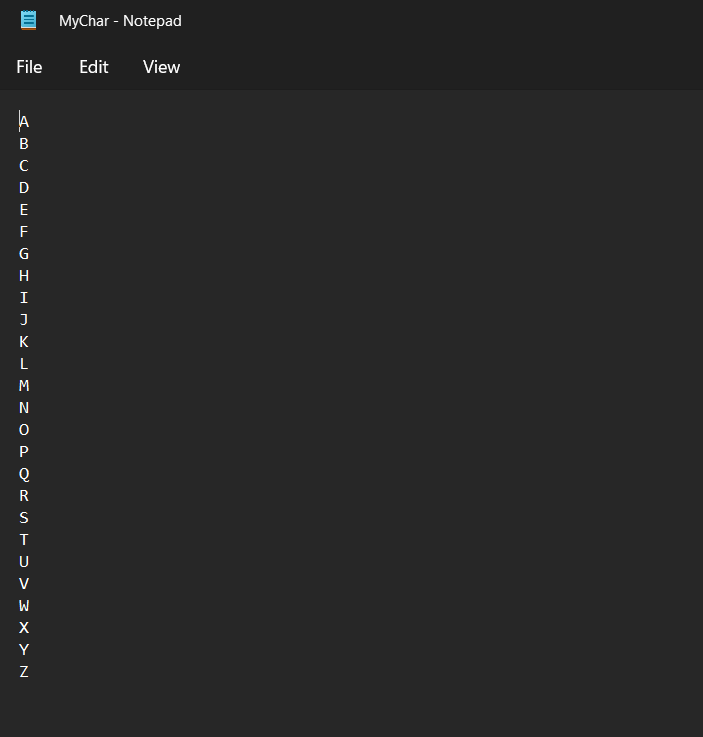
    }

    else

        inputFile.close();

}

Mychar.txt:



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 cEmployee

   {

        private:

            int empno;

            string name;

            double salary;

        public:

//              cEmployee()

//              {

//                  this->empno = 1;

//                  this->name = "Sandesh";

//                  this->salary = 40000;

//             }

               void accept(int id, string name, double salary)

               {

                    this->empno = id;

                    this->name = name;

                    this->salary = salary;

               }

               void display()

               {

                    cout<<empno<<"  "<<name<<"  "<<salary<<endl;

               }

   };

   int main()

   {

        cEmployee e1;

        e1.accept(1, "Harry", 45000);

        ofstream outfile("Employee.dat");

        outfile.write((char\*)&e1, sizeof(cEmployee));

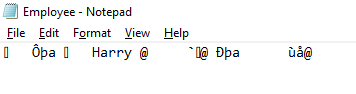
        cout<<"Employee get stored into File"<<endl;

        outfile.close();

//          ifstream inputFile("Employee.txt");

//          inputFile

   }



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()

{

    string str;

    ofstream file1("Original.txt");

    ofstream file2("New.txt");

    ifstream retrieve("Original.txt");

    if(file1.is\_open() && file2.is\_open())

    {

        cout<<"Files Created"<<endl;

        cout<<"Enter data:"<<endl;

        getline(cin, str);

        file1 << str;

        file1.close();

        while(getline(retrieve, str))

        {

            file2 << str;

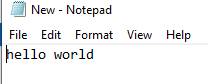
        }

    }

    file1.close();

    file2.close();

}



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> numbers;

    int no, choice;

    cout<<"Enter 1.Enter numbers 2. display 3. Sum 4. Average 5. Exit"<<endl;

    do

    {

        cout<<"Enter choice: "<<endl;

        cin>>choice;

        switch(choice)

        {

            case 1:

            {

                cout<<"Enter numbers:"<<endl;

                cin>>no;

                if(no <= 0)

                {

                    break;

                }

                numbers.push\_back(no);

                cout<<"Number added"<<endl;

                break;

            }

            case 2:

            {

                vector<int> ::iterator it1 = numbers.begin();

                vector<int> ::iterator it2 = numbers.end();

                while(it1 != it2)

                {

                    cout<<"Value: "<<\*it1<<endl;

                    it1++;

                }

                break;

            }

            case 3:

            {

                int sum = accumulate(numbers.begin(), numbers.end(), 0);

                cout<<"Sum: "<<sum<<endl;

                break;

            }

            case 4:

            {

                int sum = accumulate(numbers.begin(), numbers.end(), 0);

                double avg = sum / numbers.size();

                cout<<"Average: "<<avg<<endl;

                break;

            }

            case 5:

            {

                cout<<"Exited"<<endl;

                break;

            }

            default:

            {

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

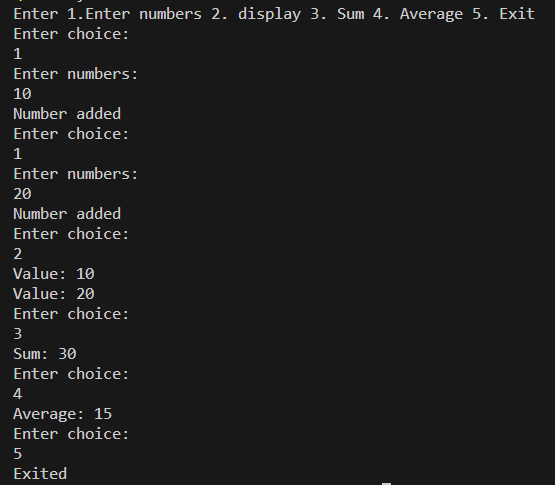
                break;

            }

        }

    }while(choice != 5);

}



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

{

    private:

        string name;

        int age;

        char grade;

    public:

        Student(string name, int age, char grade)

        {

            this->name = name;

            this->age = age;

            this->grade = grade;

        }

        void display()

        {

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

        }

};

int main()

{

    vector<Student\*> std;

    string name;

    int age, no, choice;

    char grade;

    cout<<"Enter 1.Add 2.Display 3.Exit"<<endl;

    do

    {

        cout<<"Enter choice: "<<endl;

        cin>>choice;

        switch(choice)

        {

            case 1:

            {

                cout<<"Enter name, age, grade:"<<endl;

                cin>>name>>age>>grade;

                std.push\_back(new Student(name, age, grade));

                cout<<"Added: "<<endl;

                break;

            }

            case 2:

            {

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

                {

                    std[i]->display();

                }

                break;

            }

            case 3:

            {

                cout<<"Exited"<<endl;

                break;

            }

            default:

            {

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

                break;

            }

        }

    }while(choice != 3);

}

