31: Write a program to Accept a number and display its sum of digits.:ex 568 5+6+8

#include &lt;iostream&gt;

using namespace std;

int main() {

    int num;

    cout &lt;&lt; &quot;Enter a number: &quot;;

    cin &gt;&gt; num;

    int sum = 0;

    while (num &gt; 0) {

        sum += num % 10;

        num /= 10;

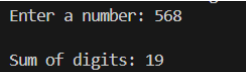
    }

    cout &lt;&lt; endl &lt;&lt; &quot;Sum of digits: &quot; &lt;&lt; sum;

    return 0;

}

Output:



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

#include &lt;iostream&gt;

using namespace std;

int main() {

    int n;

    cout &lt;&lt; &quot;Enter a number: &quot;;

    cin &gt;&gt; n;

    int sum = 0;

    for (int i = 2; i &lt;= n; i += 2) {

        sum += i;

    }

    cout &lt;&lt; endl &lt;&lt; &quot;Sum of even numbers: &quot; &lt;&lt; sum;

    int sum2 = 0;

    for (int i = 1; i &lt;= n; i += 2) {

        sum2 += i;

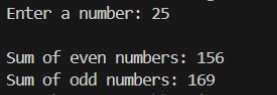
    }

    cout &lt;&lt; endl &lt;&lt; &quot;Sum of odd numbers: &quot; &lt;&lt; sum2;

    return 0;

}

Output:



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

#include &lt;iostream&gt;

using namespace std;

int main() {

    int n;

    cout &lt;&lt; &quot;Enter a number: &quot;;

    cin &gt;&gt; n;

    cout &lt;&lt; &quot;Prime numbers between 1 and &quot; &lt;&lt; n &lt;&lt; &quot; are: &quot;&lt;&lt;endl;

    for (int i = 2; i &lt;= n; i++) {

        bool isPrime = true;

        for (int j = 2; j \* j &lt;= i; j++) {

            if (i % j == 0) {

                isPrime = false;

                break;

            }

        }

        if (isPrime) {

            cout &lt;&lt; i &lt;&lt; &quot; &quot;;

        }

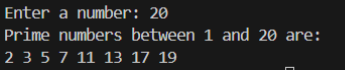
    }

    cout &lt;&lt; endl;

    return 0;

}

Output:



35:Write a program to print following pattern.

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

#include &lt;iostream&gt;

using namespace std;

int main() {

    int n;

    cout &lt;&lt; &quot;Enter the number of rows: &quot;;

    cin &gt;&gt; n;

    for (int i = 1; i &lt;= n; i++) {

        for (int j = 1; j &lt;= i; j++) {

            cout &lt;&lt; &quot;\* &quot;;

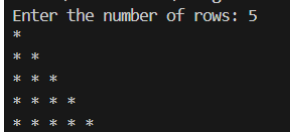
        }

        cout &lt;&lt; endl;

    }

}

Output:



36: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 &lt;iostream&gt;

using namespace std;

class Student {

public:

    void acceptInfo() {

        cout &lt;&lt; &quot;Enter roll number: &quot;;

        cin &gt;&gt; rollno;

        cout &lt;&lt; &quot;Enter marks1: &quot;;

        cin &gt;&gt; marks1;

        cout &lt;&lt; &quot;Enter marks2: &quot;;

        cin &gt;&gt; marks2;

        cout &lt;&lt; &quot;Enter marks3: &quot;;

        cin &gt;&gt; marks3;

    }

    void display() {

        int total = marks1 + marks2 + marks3;

        double percentage = static\_cast&lt;double&gt;(total) / 3.0;

        char grade;

        if (percentage &gt;= 90) {

            grade = &#39;A&#39;;

        } else if (percentage &gt;= 80) {

            grade = &#39;B&#39;;

        } else if (percentage &gt;= 70) {

            grade = &#39;C&#39;;

        } else if (percentage &gt;= 60) {

            grade = &#39;D&#39;;

        } else {

            grade = &#39;F&#39;;

        }

        cout &lt;&lt; &quot;Roll number: &quot; &lt;&lt; rollno &lt;&lt; endl;

        cout &lt;&lt; &quot;Marks1: &quot; &lt;&lt; marks1 &lt;&lt; endl;

        cout &lt;&lt; &quot;Marks2: &quot; &lt;&lt; marks2 &lt;&lt; endl;

        cout &lt;&lt; &quot;Marks3: &quot; &lt;&lt; marks3 &lt;&lt; endl;

        cout &lt;&lt; &quot;Total: &quot; &lt;&lt; total &lt;&lt; endl;

        cout &lt;&lt; &quot;Percentage: &quot; &lt;&lt; percentage &lt;&lt; &quot;%&quot; &lt;&lt; endl;

        cout &lt;&lt; &quot;Grade: &quot; &lt;&lt; grade &lt;&lt; endl;

    }

private:

    int rollno;

    int marks1, marks2, marks3;

};

int main() {

    Student s;

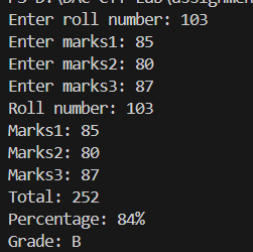
    s.acceptInfo();

    s.display();

    return 0;

}

Output:



37. 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 &lt;iostream&gt;

#include &lt;string&gt;

using namespace std;

class Person {

    private:

    string name;

    int age;

    string city;

    public:

        Person() {

            name = &quot;&quot;;

            age = 0;

            city = &quot;&quot;;

        }

        Person(string n, int a, string c) {

            name = n;

            age = a;

            city = c;

        }

        string getName() {

            return name;

        }

        int getAge() {

            return age;

        }

        string getCity() {

            return city;

        }

        void setName(string n) {

            name = n;

        }

        void setAge(int a) {

            age = a;

        }

        void setCity(string c) {

            city = c;

        }

        void display() {

            cout &lt;&lt; &quot;Name: &quot; &lt;&lt; name &lt;&lt; endl;

            cout &lt;&lt; &quot;Age: &quot; &lt;&lt; age &lt;&lt; endl;

            cout &lt;&lt; &quot;City: &quot; &lt;&lt; city &lt;&lt; endl;

        }

};

int main() {

    Person p1;

    p1.setName(&quot;Tanaya Patil&quot;);

    p1.setAge(22);

    p1.setCity(&quot;Pune&quot;);

    p1.display();

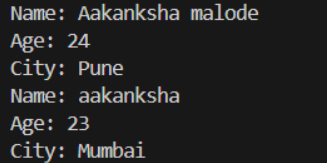
    Person p2(&quot;Tanu&quot;, 23, &quot;Jalgaon&quot;);

    p2.display();

    return 0;

}

Output:



39. 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 &lt;iostream&gt;

#include &lt;string&gt;

using namespace std;

class Date {

public:

    Date() {

        dd = 0;

        mm = 0;

yy = 0;

    }

    Date(int d, int m, int y) {

        setDate(d, m, y);

    }

    int getDd() {

        return dd;

    }

    int getMm() {

        return mm;

    }

    int getYy() {

        return yy;

    }

    void setDate(int d, int m, int y) {

        if (d &gt;= 1 &amp;&amp; d &lt;= 31) {

            dd = d;

        } else {

            cout &lt;&lt; &quot;Error: Invalid day. Please enter a day between 1 and 31.&quot;

&lt;&lt; endl;

            return;

        }

        if (m &gt;= 1 &amp;&amp; m &lt;= 12) {

            mm = m;

        } else {

            cout &lt;&lt; &quot;Error: Invalid month. Please enter a month between 1 and

12.&quot; &lt;&lt; endl;

            return;

        }

        if (y &gt;= 0) {

            yy = y;

        } else {

            cout &lt;&lt; &quot;Error: Invalid year. Please enter a year greater than or

equal to 0.&quot; &lt;&lt; endl;

            return;

        }

    }

    void setDd(int d) {

        setDate(d, mm, yy);

    }

    void setMm(int m) {

        setDate(dd, m, yy);

    }

    void setYy(int y) {

        setDate(dd, mm, y);

    }

    void display() {

        cout &lt;&lt; &quot;Date: &quot; &lt;&lt; mm &lt;&lt; &quot;/&quot; &lt;&lt; dd &lt;&lt; &quot;/&quot; &lt;&lt; yy &lt;&lt; endl;

    }

private:

    int dd;

    int mm;

    int yy;

};

int main() {

    Date d1;

    d1.setDate(25, 8, 2001);

    d1.display();

    Date d2(15, 3, 2008);

    d2.display();

    return 0;

}

Output:



40. 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>

#include <string>

using namespace std;

class Book {

private:

    string bname;

    string id;

    string author;

    float price;

public:

    Book() {

        bname = "";

        id = "";

        author = "";

        price = 0.0;

    }

    Book(string bname, string id, string author, float price) {

        this->bname = bname;

        this->id = id;

        this->author = author;

        this->price = price;

    }

    string getBname() {

        return bname;

    }

    void setBname(string bname) {

        this->bname = bname;

    }

    string getId() {

        return id;

    }

    void setId(string id) {

        this->id = id;

    }

    string getAuthor() {

        return author;

    }

    void setAuthor(string author) {

        this->author = author;

    }

    float getPrice() {

        return price;

    }

    void setPrice(float price) {

        this->price = price;

    }

void display() {

        cout << "Book Name: " << bname << endl;

        cout << "Book ID: " << id << endl;

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

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

    }

};

int main() {

    Book book1;

    book1.setBname("cPP");

    book1.setId("1");

    book1.setAuthor("james");

    book1.setPrice(29.99);

    cout << "Book Details:" << endl;

    book1.display();

cout << endl;

    Book book2("history", "5", "rock", 49.99);

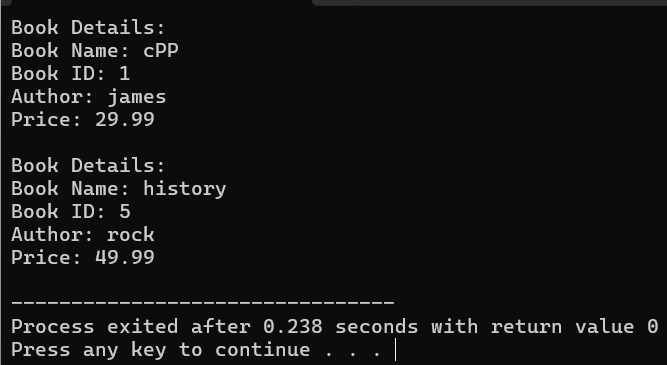
    cout << "Book Details:" << endl;

    book2.display();

    return 0;

}

Output:



41. 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 {

private:

    int x;

    int y;

public:

    Point() {

        x = 0;

        y = 0;

    }

    Point(int x, int y) {

        this->x = x;

        this->y = y;

    }

    int getX() {

        return x;

    }

    void setX(int x) {

        this->x = x;

    }

    int getY() {

        return y;

    }

    void setY(int y) {

        this->y = y;

    }

    void display() {

        cout << "Point: (" << x << ", " << y << ")" << endl;

    }

};

int main() {

    Point point1;

    point1.setX(5);

    point1.setY(10);

    cout << "Point 1 Details:" << endl;

    point1.display();

    cout << endl;

    Point point2(3, 7);

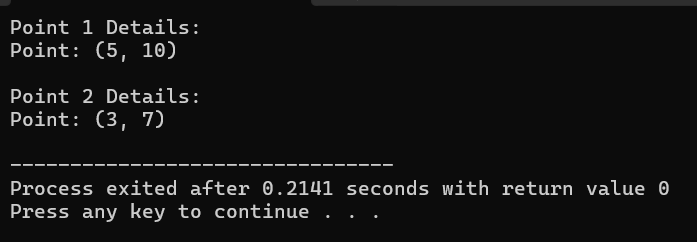
    cout << "Point 2 Details:" << endl;

    point2.display();

    return 0;

}

Output:



42. 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>

using namespace std;

class ComplexNumber {

private:

    float real;

    float imaginary;

public:

    ComplexNumber() {

        real = 0.0;

        imaginary = 0.0;

    }

    ComplexNumber(float real, float imaginary) {

        this->real = real;

        this->imaginary = imaginary;

    }

    float getReal() {

        return real;

    }

    void setReal(float real) {

        this->real = real;

    }

    float getImaginary() {

        return imaginary;

    }

    void setImaginary(float imaginary) {

        this->imaginary = imaginary;

    }

    void display() {

        cout << "Complex Number: " << real << " + " << imaginary << "i" << endl;

    }

};

int main() {

    ComplexNumber complex1;

    complex1.setReal(12.5);

    complex1.setImaginary(2.0);

    cout << "Complex 1 Details:" << endl;

    complex1.display();

    cout << endl;

    ComplexNumber complex2(1.7, 4.2);

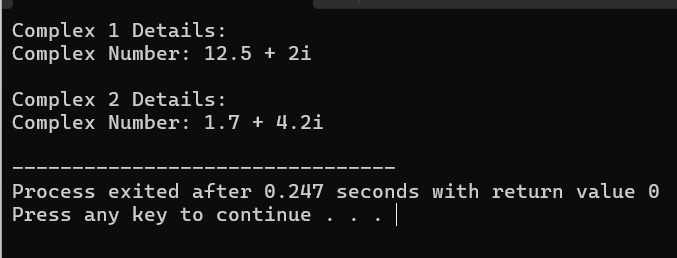
    cout << "Complex 2 Details:" << endl;

    complex2.display();

    return 0;

}

Output:



43 Solve this.

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

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

43.1 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 <vector>

using namespace std;

class Employee {

protected:

    int id;

    string name;

    int deptId;

    double basicSalary;

public:

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

        : id(id), name(name), deptId(deptId), basicSalary(basicSalary) {}

    virtual double computeNetSalary() {

        return 0;

    }

    virtual void display() {

        cout << "ID: " << id << ", Name: " << name << ", Department ID: " << deptId << ", Basic 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), perfBonus(perfBonus) {}

    double computeNetSalary() override {

        return basicSalary + perfBonus;

    }

    void display() override {

        Employee::display();

        cout << "Performance Bonus: " << perfBonus << endl;

    }

};

class Worker : public Employee {

private:

    int hoursWorked;

    double hourlyRate;

public:

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

        : Employee(id, name, deptId, basicSalary), hoursWorked(hoursWorked), hourlyRate(hourlyRate) {}

    double computeNetSalary() override {

        return basicSalary + (hoursWorked \* hourlyRate);

    }

    double getHourlyRate() {

        return hourlyRate;

    }

    void display() override {

        Employee::display();

        cout << "Hours Worked: " << hoursWorked << ", Hourly Rate: " << hourlyRate << endl;

    }

};

int main() {

    vector<Employee\*> employees;

    char choice;

    do {

        cout << "1. Hire Manager" << endl;

        cout << "2. Hire Worker" << endl;

        cout << "3. Display information of all employees net salary" << endl;

        cout << "4. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch(choice) {

            case '1': {

                int id, deptId;

                string name;

                double basicSalary, perfBonus;

                cout << "Enter Manager details:" << endl;

                cout << "ID: ";

                cin >> id;

                cout << "Name: ";

                cin >> name;

                cout << "Department ID: ";

                cin >> deptId;

                cout << "Basic Salary: ";

                cin >> basicSalary;

                cout << "Performance Bonus: ";

                cin >> perfBonus;

                employees.push\_back(new Manager(id, name, deptId, basicSalary, perfBonus));

                break;

            }

            case '2': {

                int id, deptId, hoursWorked;

                string name;

                double basicSalary, hourlyRate;

                cout << "Enter Worker details:" << endl;

                cout << "ID: ";

                cin >> id;

                cout << "Name: ";

                cin >> name;

                cout << "Department ID: ";

                cin >> deptId;

                cout << "Basic Salary: ";

                cin >> basicSalary;

                cout << "Hours Worked: ";

                cin >> hoursWorked;

                cout << "Hourly Rate: ";

                cin >> hourlyRate;

                employees.push\_back(new Worker(id, name, deptId, basicSalary, hoursWorked, hourlyRate));

                break;

            }

            case '3':

                cout << "Employees Information:" << endl;

                for (Employee\* emp : employees) {

                    emp->display();

                    cout << "Net Salary: " << emp->computeNetSalary() << endl;

                }

                break;

            case '4':

                cout << "Exiting program." << endl;

                break;

            default:

                cout << "Invalid choice. Please enter again." << endl;

        }

    } while (choice != '4');

    for (Employee\* emp : employees) {

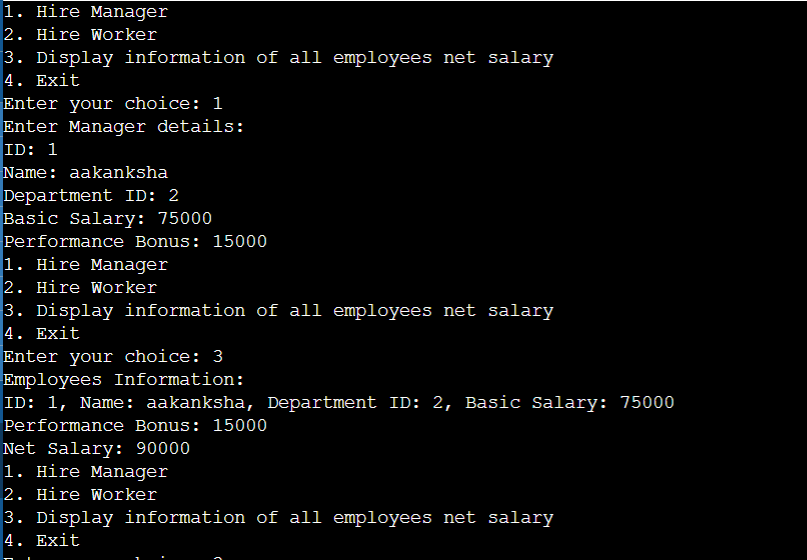
        delete emp;

    }

    return 0;

}

Output:



44: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 <string>

using namespace std;

class BankAccount {

private:

    int acctNo;

    string customerName;

    double balance;

public:

    BankAccount(int acctNo, string customerName, double balance) {

        this->acctNo = acctNo;

        this->customerName = customerName;

        this->balance = balance;

    }

    void withdraw(double amt) {

        if (amt > 0 && amt <= balance) {

            balance -= amt;

            cout << "Withdrawal successful. Current balance: " << balance << endl;

        } else {

            cout << " insufficient balance." << endl;

        }

    }

    void deposit(double amt) {

        if (amt > 0) {

            balance += amt;

            cout << "Deposit successful. balance: " << balance << endl;

        } else {

            cout << "Invalid  amount." << endl;

        }

    }

};

int main() {

    BankAccount account(123456, "suresh ", 1000.0);

    cout << "Initial balance: 1000.0" << endl;

    account.withdraw(500.0);

    account.deposit(200.0);

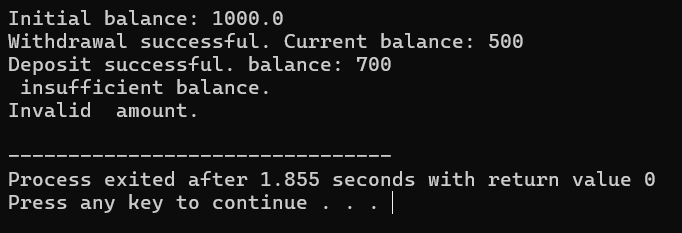
    account.withdraw(1500.0);

    account.deposit(-10.0);

    return 0;

}

Output :



45: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 double area() const = 0;

};

class Rectangle : public Shape {

private:

    double length;

    double width;

public:

    Rectangle(double l, double w) : length(l), width(w) {}

    double area() const override {

        return length \* width;

    }

};

class Circle : public Shape {

private:

    double radius;

public:

    Circle(double r) : radius(r) {}

    double area() const override {

        return 3.14 \* radius \* radius;

    }

};

class Square : public Shape {

private:

    double side;

public:

    Square(double s) : side(s) {}

    double area() const override {

        return side \* side;

    }

};

int main() {

    Rectangle rectangle(5, 4);

    Circle circle(3);

    Square square(6);

    cout << "Area of Rectangle: " << rectangle.area() << endl;

    cout << "Area of Circle: " << circle.area() << endl;

    cout << "Area of Square: " << square.area() << endl;

    return 0;

}

Output:

