main.cpp

Share Run

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
#include<bits/stdc++.h>
using namespace std;
void sum(int n)
{
    if(n>0){
    int sum = n*(n+1)/2;
    cout<<"sum of n natural numbers: "<<sum<<endl;
    }
}

int main() {
    int n;
    cout<<"enter the number "<<endl;
    cin>>n;
    sum(n);

    return 0;
}
```

Output

Clear

```
enter the number
5
sum of n natural numbers: 15


=== Code Execution Successful ===
```

main.cpp

Share Run

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
#include<bits/stdc++.h>
using namespace std;
void count_digits(int n)
{
  int count=0;
  while(n!=0)
  {
      n=n/10;
      count++;
  }
  cout<<count;
}

int main() {
    int n;
    cout<<"enter the number "<<endl;
    cin>>n;
    count_digits(n);

    return 0;
}
```

Output

Clear

```
enter the number
12345
5

=== Code Execution Successful ===
```

main.cpp

Share Run

```cpp
#include <iostream>
#include <cmath>
double area(double radius) {
    return M_PI * radius * radius;
}
double area(double length, double width) {
    return length * width;
}
double area(double base, double height, bool isTriangle) {
    if (isTriangle) {
        return 0.5 * base * height;
    }
    return 0.0;
}
int main() {
    double radius;
    std::cout << "Enter the radius of the circle: ";
    std::cin >> radius;
    std::cout << "Area of the circle: " << area(radius) << std::endl;
    double length, width;
    std::cout << "\nEnter the length and width of the rectangle: ";
    std::cin >> length >> width;
    std::cout << "Area of the rectangle: " << area(length, width) << std::endl;
    double base, height;
    std::cout << "\nEnter the base and height of the triangle: ";
    std::cin >> base >> height;
```

Output

Clear

```
Enter the radius of the circle: 5
Area of the circle: 78.5398

Enter the length and width of the rectangle: 4
6
Area of the rectangle: 24

Enter the base and height of the triangle: 3
7
Area of the triangle: 10.5


=== Code Execution Successful ===
```

main.cpp

Share Run

```cpp
    return length * width;
}
double area(double base, double height, bool isTriangle) {
    if (isTriangle) {
        return 0.5 * base * height;
    }
    return 0.0;
}
int main() {
    double radius;
    std::cout << "Enter the radius of the circle: ";
    std::cin >> radius;
    std::cout << "Area of the circle: " << area(radius) << std::endl;
    double length, width;
    std::cout << "\nEnter the length and width of the rectangle: ";
    std::cin >> length >> width;
    std::cout << "Area of the rectangle: " << area(length, width) << std::endl;
    double base, height;
    std::cout << "\nEnter the base and height of the triangle: ";
    std::cin >> base >> height;
    std::cout << "Area of the triangle: " << area(base, height, true) << std::endl;

    return 0;
}
```

Output

Clear

```
Enter the radius of the circle: 5
Area of the circle: 78.5398

Enter the length and width of the rectangle: 4
6
Area of the rectangle: 24

Enter the base and height of the triangle: 3
7
Area of the triangle: 10.5


=== Code Execution Successful ===
```

main.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

// Base class: Account
class Account {
protected:
    double balance;

public:
    Account(double bal) : balance(bal) {}
    virtual void calculateInterest() = 0; // Pure virtual function
    virtual void display() {
        cout << "Balance: " << balance << endl;
    }
    virtual ~Account() {}
};

// Derived class: SavingsAccount
class SavingsAccount : public Account {
    double rate;
    int time;

public:
    SavingsAccount(double bal, double r, int t) : Account(bal), rate(r), time(t) {}

    void calculateInterest() override {
        double interest = balance * rate * time;
```

Output

```
Savings Account Interest: 100
Savings Account Balance: 1100
No interest for Current Account. Deducting maintenance fee: 50
Current Account Balance: 1950

=== Code Execution Successful ===
```

main.cpp

Share Run

```cpp
        cout << "Savings Account Interest: " << interest << endl;
        balance += interest;
    }

    void display() override {
        cout << "Savings Account Balance: " << balance << endl;
    }
};

// Derived class: CurrentAccount
class CurrentAccount : public Account {
    double maintenanceFee;

public:
    CurrentAccount(double bal, double fee) : Account(bal), maintenanceFee(fee) {}

    void calculateInterest() override {
        cout << "No interest for Current Account. Deducting maintenance fee: " << maintenanceFee << endl;
        balance -= maintenanceFee;
    }

    void display() override {
        cout << "Current Account Balance: " << balance << endl;
    }
};

int main() {
```

```
int main() {
    Account* accounts[2];

    // Create SavingsAccount and CurrentAccount
    accounts[0] = new SavingsAccount(1000.0, 0.05, 2);
    accounts[1] = new CurrentAccount(2000.0, 50.0);

    // Calculate interest and display details
    for (int i = 0; i < 2; i++) {
        accounts[i]->calculateInterest();
        accounts[i]->display();
        delete accounts[i];
    }

    return 0;
}
```

main.cpp

Share Run

```cpp
#include <iostream>
#include <string>
using namespace std;

// Base class: Employee
class Employee {
protected:
    string name;
    int id;
    double salary;

public:
    Employee(string n, int i, double s) : name(n), id(i), salary(s) {}

    virtual double calculateEarnings() = 0; // Pure virtual function

    virtual void display() {
        cout << "Name: " << name << ", ID: " << id << ", Base Salary: " << salary << endl;
    }

    virtual ~Employee() {}
};

// Derived class: Manager
class Manager : public Employee {
    double bonus;

```

Output

Clear

```
Name: Alice, ID: 101, Base Salary: 50000
Bonus: 10000, Total Earnings: 60000
Name: Bob, ID: 102, Base Salary: 40000
Overtime Hours: 20, Overtime Rate: 50, Total Earnings: 41000


=== Code Execution Successful ===
```

```
    Manager(string n, int i, double s, double b) : Employee(n, i, s), bonus(b) {}

    double calculateEarnings() override {
        return salary + bonus;
    }

    void display() override {
        Employee::display();
        cout << "Bonus: " << bonus << ", Total Earnings: " << calculateEarnings() << endl;
    }
};

// Derived class: Developer
class Developer : public Employee {
    double overtimeHours;
    double overtimeRate;

public:
    Developer(string n, int i, double s, double hours, double rate) : Employee(n, i, s), overtimeHours(hours), overtimeRate(rate) {}

    double calculateEarnings() override {
        return salary + (overtimeHours * overtimeRate);
    }

    void display() override {
        Employee::display();
```

```
        Employee::display();
        cout << "Overtime Hours: " << overtimeHours << ", Overtime Rate: " << overtimeRate
            << ", Total Earnings: " << calculateEarnings() << endl;
    }
};

int main() {
    Employee* employees[2];

    // Create Manager and Developer objects
    employees[0] = new Manager("Alice", 101, 50000, 10000);
    employees[1] = new Developer("Bob", 102, 40000, 20, 50);

    // Display employee details and calculate earnings
    for (int i = 0; i < 2; i++) {
        employees[i]->display();
        delete employees[i];
    }

    return 0;
}
```