Name : Archi Bansal Section : IOT-620

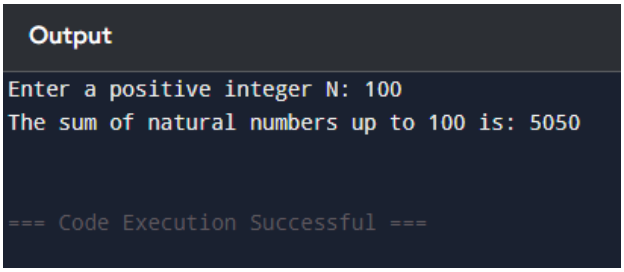
UID: 22BCS15264 Date: 19/12/2024

Domain Winning Winter Camp

1) Sum of Natural Numbers up to N

Code:

```
#include <iostream>
using namespace std;
int main() {
   int N;
   cout << "Enter a positive integer N: ";
   cin >> N;
   int sum = N * (N + 1) / 2;
   cout << "The sum of natural numbers up to " << N << " is: " << sum << endl;
   return 0;
}
```

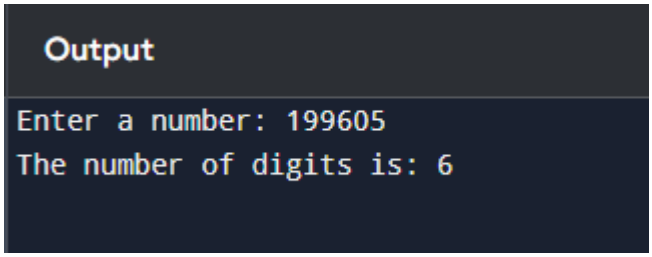


2) Count Digits in a Number

Code:

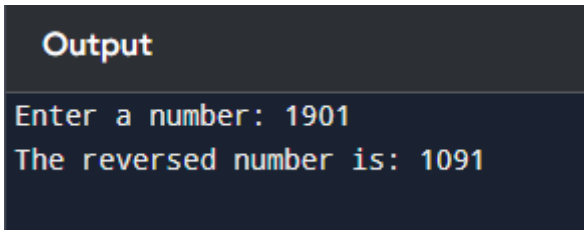
```
#include <iostream>
using namespace std;
int main() {
   int num, count = 0;
   cout << "Enter a number: ";
   cin >> num;
   if (num < 0) {
      num = -num;
   }
   do {
      count++;
      num /= 10;
   } while (num != 0);

   cout << "The number of digits is: " << count << endl;
   return 0;
}
```



3) Reverse a Number
Code:

```
#include <iostream>
using namespace std;
int main() {
   int num, reversed = 0;
   cout << "Enter a number: ";
   cin >> num;
   bool isNegative = false;
   if (num < 0) {
      isNegative = true;
      num = -num;
   }
   while (num != 0) {
      int digit = num % 10;
      reversed = reversed * 10 + digit;
      num /= 10;
   }
   if (isNegative) {
      reversed = -reversed;
   }
   cout << "The reversed number is: " << reversed << endl;
   return 0;
}
```



4) Function Overloading for Calculating Area.
Code:

```
#include <iostream>

using namespace std;

double calculateArea(double radius) {

   return 3.14159 * radius * radius; // Circle

}
```

```
double calculateArea(double length, double width) {
    return length * width; // Rectangle
}
double calculateArea(double base, double height, int) {
    return 0.5 * base * height; // Triangle
}
int main() {
    cout << "Choose an option to calculate area:\n1. Circle\n2. Rectangle\n3. Triangle\n";
    int choice;
    cin >> choice;

    if (choice == 1) {
        double radius;
        cout << "Enter the radius: ";
        cin >> radius;
        cout << "Area of the circle: " << calculateArea(radius) << endl;
    } else if (choice == 2) {
        double length, width;
        cout << "Enter length and width: ";
        cin >> length >> width;
        cout << "Area of the rectangle: " << calculateArea(length, width) << endl;
    } else if (choice == 3) {
        double base, height;
        cout << "Enter base and height: ";
        cin >> base >> height;
        cout << "Area of the triangle: " << calculateArea(base, height, 0) << endl;
    } else {
        cout << "Invalid choice!" << endl;
    }

    return 0;
}
```

```
Output
Choose an option to calculate area:
1. Circle
2. Rectangle
3. Triangle
2
Enter length and width:
20 10
Area of the rectangle: 200
```

5) Matrix Multiplication Using Function Overloading

Code:

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

// Function to multiply two matrices (rowsA x colsA) and (rowsB x colsB)
void multiplyMatrix(int rowsA, int colsA, int A[][10], int rowsB, int colsB, int B[][10], int result[][10])
{
    if (colsA != rowsB) {
        cout << "Matrix multiplication not possible. Column count of A must equal row count of B." <<
endl;
        return;
    }
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsB; j++) {
            result[i][j] = 0;
            for (int k = 0; k < colsA; k++) {
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

// Function to display a matrix
void displayMatrix(int rows, int cols, int matrix[][10]) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    int A[10][10], B[10][10], result[10][10];
    int rowsA, colsA, rowsB, colsB;

    cout << "Enter rows and columns for matrix A: ";
    cin >> rowsA >> colsA;
```

```
    cout << "Enter elements of matrix A:" << endl;
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsA; j++) {
            cin >> A[i][j];
        }
    }

    cout << "Enter rows and columns for matrix B: ";
    cin >> rowsB >> colsB;

    cout << "Enter elements of matrix B:" << endl;
    for (int i = 0; i < rowsB; i++) {
        for (int j = 0; j < colsB; j++) {
            cin >> B[i][j];
        }
    }

    cout << "Result of matrix multiplication:" << endl;
    multiplyMatrix(rowsA, colsA, A, rowsB, colsB, B, result);
    displayMatrix(rowsA, colsB, result);

    return 0;
}
```

Output

```
Enter rows and columns for matrix A: 2 2
Enter elements of matrix A:
1 2 3 4
Enter rows and columns for matrix B: 2 2
Enter elements of matrix B:
4 5 6 7
Result of matrix multiplication:
16 19
36 43
```

6) Hierarchical Inheritance for Employee Management System Objective

Code:

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

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

public:
    void inputBasicDetails() {
        cout << "Enter Employee Name: ";
        cin >> name;
```

```
        cout << "Enter Employee ID: ";
        cin >> id;
        cout << "Enter Basic Salary: ";
        cin >> salary;
    }

    virtual void displayDetails() const {
        cout << "Name: " << name << endl;
        cout << "ID: " << id << endl;
        cout << "Basic Salary: " << salary << endl;
    }

    virtual double calculateTotalEarnings() const = 0; // Pure virtual function
};

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

public:
    void inputManagerDetails() {
        inputBasicDetails();
        cout << "Enter Performance Bonus: ";
        cin >> bonus;
    }

    double calculateTotalEarnings() const override {
        return salary + bonus;
    }

    void displayDetails() const override {
        Employee::displayDetails();
        cout << "Performance Bonus: " << bonus << endl;
        cout << "Total Earnings: " << calculateTotalEarnings() << endl;
    }
};

// Derived class for Developer
class Developer : public Employee {
private:
    double overtimeRate;
    int extraHours;

public:
    void inputDeveloperDetails() {
        inputBasicDetails();
        cout << "Enter Overtime Rate: ";
```

```
        cin >> overtimeRate;
        cout << "Enter Extra Hours Worked: ";
        cin >> extraHours;
    }

    double calculateTotalEarnings() const override {
        return salary + (overtimeRate * extraHours);
    }

    void displayDetails() const override {
        Employee::displayDetails();
        cout << "Overtime Rate: " << overtimeRate << endl;
        cout << "Extra Hours Worked: " << extraHours << endl;
        cout << "Total Earnings: " << calculateTotalEarnings() << endl;
    }
};

int main() {
    Manager mgr;
    Developer dev;

    cout << "Enter Manager Details:" << endl;
    mgr.inputManagerDetails();

    cout << "\nEnter Developer Details:" << endl;
    dev.inputDeveloperDetails();

    cout << "\nManager Details:" << endl;
    mgr.displayDetails();

    cout << "\nDeveloper Details:" << endl;
    dev.displayDetails();

    return 0;
}
```

Output

```
Enter Manager Details:
Enter Employee Name: archi
Enter Employee ID: 1901
Enter Basic Salary: 50000
Enter Performance Bonus: 10000

Enter Developer Details:
Enter Employee Name: rohit
Enter Employee ID: 1996
Enter Basic Salary: 60000
Enter Overtime Rate: 20000
Enter Extra Hours Worked: 2

Manager Details:
Name: archi
ID: 1901
Basic Salary: 50000
Performance Bonus: 10000
Total Earnings: 60000

Developer Details:
Name: rohit
ID: 1996
Basic Salary: 60000
Overtime Rate: 20000
Extra Hours Worked: 2
Total Earnings: 100000
```