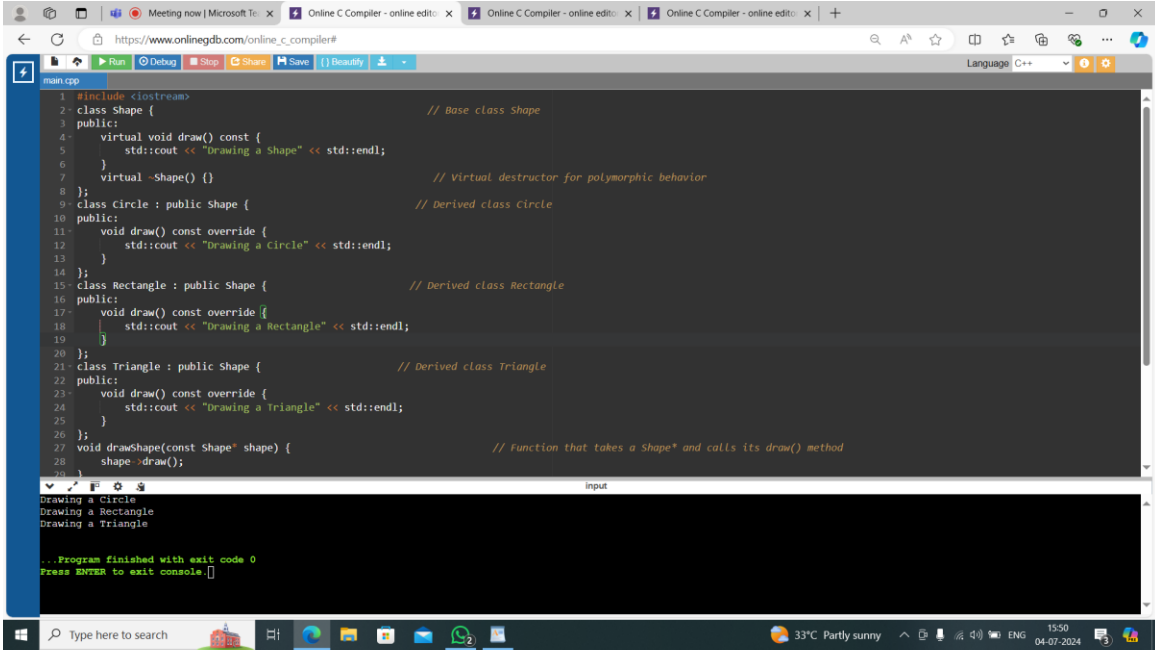
Polymorphism:

Design a class hierarchy for a simple graphic editor with base class Shape and derived classes Circle, Rectangle, and Triangle. Implement a virtual function draw() in the base class and override it in the derived classes. Write a function that takes a Shape* and calls its draw() method.

```
#include <iostream>

class Shape {                                                    // Base class Shape

public:

    virtual void draw() const {

        std::cout << "Drawing a Shape" << std::endl;

    }

    virtual ~Shape() {}                                         // Virtual destructor for polymorphic behavior

};

class Circle : public Shape {                            // Derived class Circle

public:

    void draw() const override {

        std::cout << "Drawing a Circle" << std::endl;

    }

};

class Rectangle : public Shape {                        // Derived class Rectangle

public:

    void draw() const override {

        std::cout << "Drawing a Rectangle" << std::endl;

    }

};

class Triangle : public Shape {                      // Derived class Triangle

public:

    void draw() const override {
        std::cout << "Drawing a Triangle" << std::endl;

    }

};

void drawShape(const Shape* shape) {                                        // Function that takes a Shape* and calls its draw() method

    shape->draw();

}

int main() {                                        // Main function to test the functionality

    Circle circle;

    Rectangle rectangle;

    Triangle triangle;

    drawShape(&circle);                        // Polymorphic behavior through pointers

    drawShape(&rectangle);

    drawShape(&triangle);

    return 0;

}
```

OUTPUT :-



Meeting now | Microsoft Te
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
https://www.onlinegdb.com/online_c_compiler#
Run
Debug
Stop
Share
Save
{} Beautify
Language C++
main.cpp
input
Type here to search
33°C Partly sunny
ENG
15:50
04-07-2024

```
#include <iostream>
class Shape {                                    // Base class Shape
public:
    virtual void draw() const {
        std::cout << "Drawing a Shape" << std::endl;
    }
    virtual ~Shape() {}                              // Virtual destructor for polymorphic behavior
};
class Circle : public Shape {                    // Derived class Circle
public:
    void draw() const override {
        std::cout << "Drawing a Circle" << std::endl;
    }
};
class Rectangle : public Shape {                 // Derived class Rectangle
public:
    void draw() const override {
        std::cout << "Drawing a Rectangle" << std::endl;
    }
};
class Triangle : public Shape {                  // Derived class Triangle
public:
    void draw() const override {
        std::cout << "Drawing a Triangle" << std::endl;
    }
};
void drawShape(const Shape* shape) {                              // Function that takes a Shape* and calls its draw() method
    shape->draw();
```

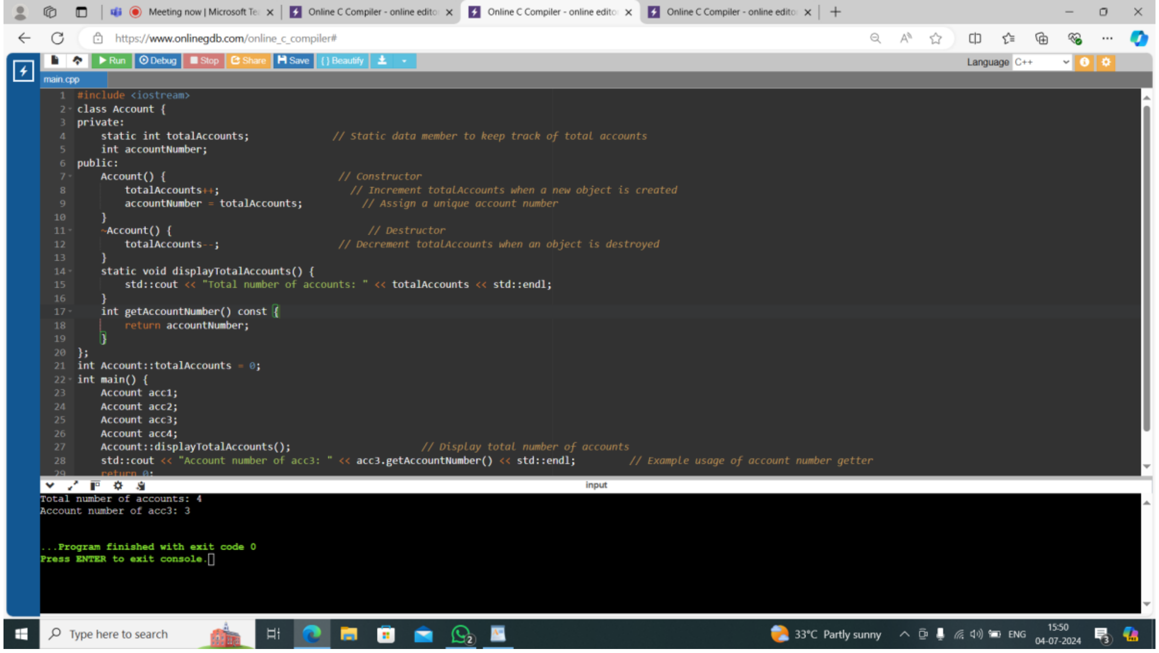
```
Drawing a Circle
Drawing a Rectangle
Drawing a Triangle


...Program finished with exit code 0
Press ENTER to exit console.
```

Static Members:

Create a class Account that has a static data member totalAccounts to keep track of the number of accounts created. Implement necessary constructors and destructors to update totalAccounts. Write a function to display the total number of accounts.

```
#include <iostream>
class Account {
private:
    static int totalAccounts;                 // Static data member to keep track of total accounts
    int accountNumber;
public:
    Account() {                                // Constructor
        totalAccounts++;                       // Increment totalAccounts when a new object is created
        accountNumber = totalAccounts;         // Assign a unique account number
    }
    ~Account() {                               // Destructor
        totalAccounts--;                       // Decrement totalAccounts when an object is destroyed
    }
    static void displayTotalAccounts() {
        std::cout << "Total number of accounts: " << totalAccounts << std::endl;
    }
    int getAccountNumber() const {
        return accountNumber;
    }
};
int Account::totalAccounts = 0;
int main() {
    Account acc1;

    Account acc2;

    Account acc3;

    Account acc4;

    Account::displayTotalAccounts();                              // Display total number of accounts

    std::cout << "Account number of acc3: " << acc3.getAccountNumber() << std::endl;                    //
Example usage of account number getter

    return 0;

}
```

OUTPUT :-



Meeting now | Microsoft Te
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
https://www.onlinegdb.com/online_c_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
input
Type here to search
33°C Partly sunny
ENG
15:50
04-07-2024

```cpp
#include <iostream>
class Account {
private:
    static int totalAccounts;              // Static data member to keep track of total accounts
    int accountNumber;
public:
    Account() {                             // Constructor
        totalAccounts++;                      // Increment totalAccounts when a new object is created
        accountNumber = totalAccounts;          // Assign a unique account number
    }
    ~Account() {                                // Destructor
        totalAccounts--;                    // Decrement totalAccounts when an object is destroyed
    }
    static void displayTotalAccounts() {
        std::cout << "Total number of accounts: " << totalAccounts << std::endl;
    }
    int getAccountNumber() const {
        return accountNumber;
    }
};
int Account::totalAccounts = 0;
int main() {
    Account acc1;
    Account acc2;
    Account acc3;
    Account acc4;
    Account::displayTotalAccounts();                    // Display total number of accounts
    std::cout << "Account number of acc3: " << acc3.getAccountNumber() << std::endl;         // Example usage of account number getter
    return 0;
```

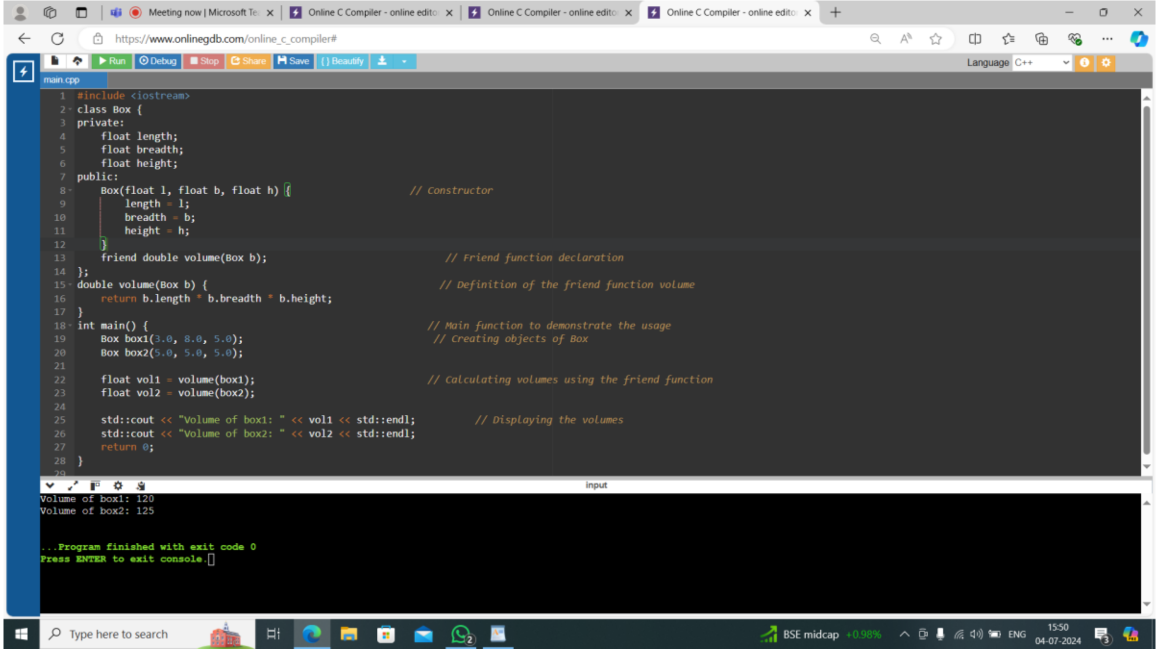
```
Total number of accounts: 4
Account number of acc3: 3


...Program finished with exit code 0
Press ENTER to exit console.
```

Friend Functions:

Implement a class Box that has private data members length, breadth, and height. Write a friend function volume() that calculates and returns the volume of the box. Create objects of Box and use the friend function to compute their volumes.

```
#include <iostream>

class Box {

private:
        float length;
        float breadth;
        float height;
public:
        Box(float l, float b, float h) {                        // Constructor
                length = l;
                breadth = b;
                height = h;
        }
        friend double volume(Box b);                                  // Friend function declaration
};
double volume(Box b) {                                                // Definition of the friend function volume
        return b.length * b.breadth * b.height;
}
int main() {                                                          // Main function to demonstrate the usage
        Box box1(3.0, 8.0, 5.0);                                      // Creating objects of Box
        Box box2(5.0, 5.0, 5.0);

        float vol1 = volume(box1);                                    // Calculating volumes using the friend function
        float vol2 = volume(box2);

        std::cout << "Volume of box1: " << vol1 << std::endl;         // Displaying the volumes
        std::cout << "Volume of box2: " << vol2 << std::endl;
        return 0;
}
```

OUTPUT :-



Meeting now | Microsoft Te
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
https://www.onlinegdb.com/online_c_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
input
Type here to search
BSE midcap +0.98%
ENG
15:50
04-07-2024

```
#include <iostream>
class Box {
private:
    float length;
    float breadth;
    float height;
public:
    Box(float l, float b, float h) {                    // Constructor
        length = l;
        breadth = b;
        height = h;
    }
    friend double volume(Box b);                            // Friend function declaration
};
double volume(Box b) {                                     // Definition of the friend function volume
    return b.length * b.breadth * b.height;
}
int main() {                                              // Main function to demonstrate the usage
    Box box1(3.0, 8.0, 5.0);                               // Creating objects of Box
    Box box2(5.0, 5.0, 5.0);

    float vol1 = volume(box1);                            // Calculating volumes using the friend function
    float vol2 = volume(box2);

    std::cout << "Volume of box1: " << vol1 << std::endl;         // Displaying the volumes
    std::cout << "Volume of box2: " << vol2 << std::endl;
    return 0;
}
```

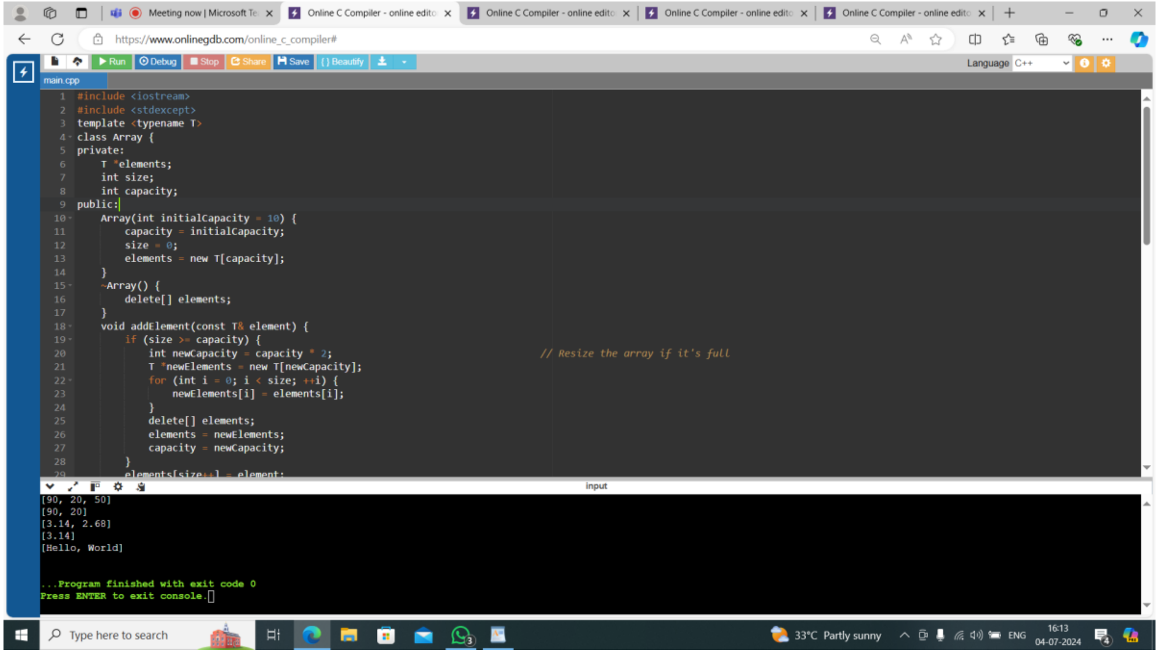
```
Volume of box1: 120
Volume of box2: 125


...Program finished with exit code 0
Press ENTER to exit console.
```

Templates:

Write a template class Array that can store an array of any data type. Include member functions to perform operations like adding an element, removing an element, and displaying the array. Demonstrate the functionality with different data types.

```
#include <iostream>

#include <stdexcept>

template <typename T>

class Array {

private:

    T *elements;

    int size;

    int capacity;

public:

    Array(int initialCapacity = 10) {

        capacity = initialCapacity;

        size = 0;

        elements = new T[capacity];
    }
    ~Array() {
        delete[] elements;
    }
    void addElement(const T& element) {
        if (size >= capacity) {
            int newCapacity = capacity * 2;                                        // Resize the
array if it's full
            T *newElements = new T[newCapacity];
            for (int i = 0; i < size; ++i) {
                newElements[i] = elements[i];
            }
            delete[] elements;
            elements = newElements;
            capacity = newCapacity;
        }
        elements[size++] = element;
    }
    void removeElement(int index) {
        if (index < 0 || index >= size) {
            throw std::out_of_range("Index out of range");
        }
        for (int i = index; i < size - 1; ++i) {
            elements[i] = elements[i + 1];
        }
        size--;
    }
    void display() const {
        std::cout << "[";
        for (int i = 0; i < size; ++i) {
            std::cout << elements[i];
            if (i < size - 1) {
                std::cout << ", ";
            }
        }
        std::cout << "]" << std::endl;
    }
};
int main() {
    Array<int> intArray;                    // Example usage with different data types
    intArray.addElement(90);                    // Integer array
    intArray.addElement(20);
    intArray.addElement(50);
    intArray.display();
    intArray.removeElement(2);
    intArray.display();
    Array<double> doubleArray;                   // Double array
    doubleArray.addElement(3.14);
    doubleArray.addElement(2.68);
    doubleArray.display();
    doubleArray.removeElement(1);
    doubleArray.display();
    Array<std::string> stringArray;         // String array
    stringArray.addElement("Hello");
    stringArray.addElement("World");
    stringArray.display();
    return 0;
}
```

OUTPUT :-



```
#include <iostream>
#include <stdexcept>
template <typename T>
class Array {
private:
    T *elements;
    int size;
    int capacity;
public:
    Array(int initialCapacity = 10) {
        capacity = initialCapacity;
        size = 0;
        elements = new T[capacity];
    }
    ~Array() {
        delete[] elements;
    }
    void addElement(const T& element) {
        if (size >= capacity) {
            int newCapacity = capacity * 2;                    // Resize the array if it's full
            T *newElements = new T[newCapacity];
            for (int i = 0; i < size; ++i) {
                newElements[i] = elements[i];
            }
            delete[] elements;
            elements = newElements;
            capacity = newCapacity;
        }
        elements[size++] = element;
```

```
[90, 20, 50]
[90, 20]
[3.14, 2.68]
[3.14]
[Hello, World]


...Program finished with exit code 0
Press ENTER to exit console.
```

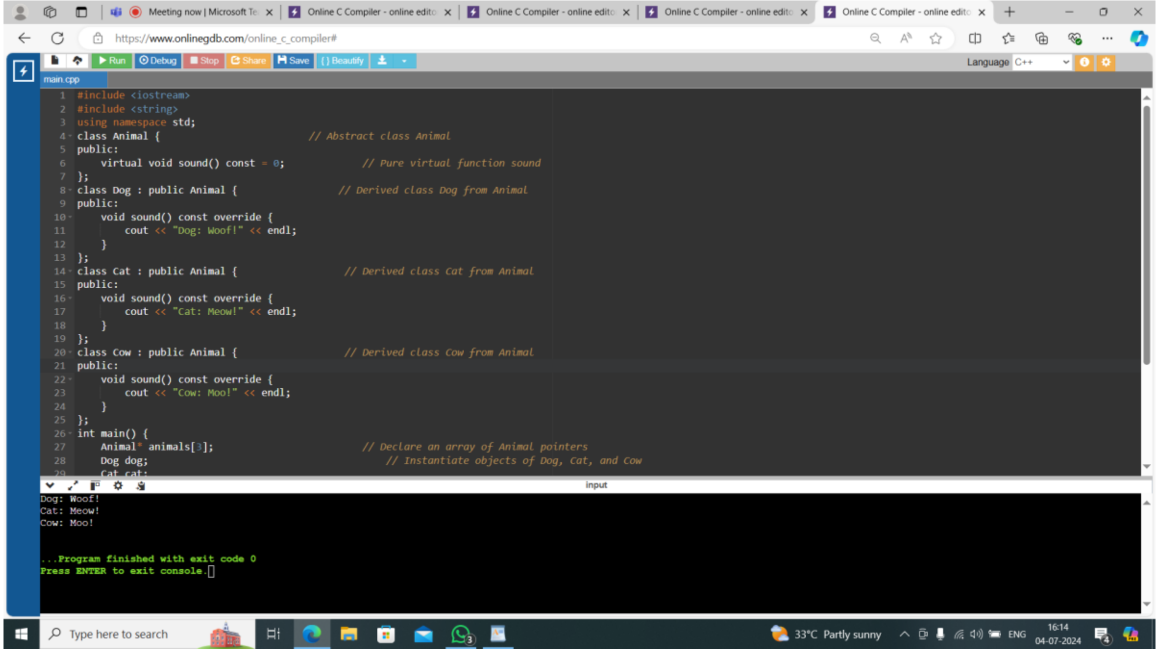
Polymorphism with Abstract Classes:

Create an abstract class Animal with a pure virtual function sound(). Derive classes Dog, Cat, and Cow from Animal and override the sound() function in each derived class. Write a program to demonstrate polymorphism using these classes.

```
#include <iostream>

#include <string>

using namespace std;

class Animal {                                  // Abstract class Animal

public:

    virtual void sound() const = 0;                        // Pure virtual function sound

};

class Dog : public Animal {                        // Derived class Dog from Animal

public:

    void sound() const override {
        cout << "Dog: Woof!" << endl;
    }
};
class Cat : public Animal {                         // Derived class Cat from Animal
public:
    void sound() const override {
        cout << "Cat: Meow!" << endl;
    }
};
class Cow : public Animal {                         // Derived class Cow from Animal
public:
    void sound() const override {
        cout << "Cow: Moo!" << endl;
    }
};
int main() {
    Animal* animals[3];                                     // Declare an array of Animal pointers
    Dog dog;                                                        // Instantiate objects of Dog, Cat, and
Cow
    Cat cat;
    Cow cow;
    animals[0] = &dog;                                      // Assign addresses of objects to Animal
pointers
    animals[1] = &cat;
    animals[2] = &cow;
    for (int i = 0; i < 3; ++i) {                                   // Use polymorphism to call the sound()
function
        animals[i]->sound();
    }
```

```
        return 0;

}
```

OUTPUT :-



Meeting now | Microsoft Te
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
https://www.onlinegdb.com/online_c_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
input
Type here to search
33°C Partly sunny
ENG
16:14
04-07-2024

```
#include <iostream>
#include <string>
using namespace std;
class Animal {                              // Abstract class Animal
public:
    virtual void sound() const = 0;             // Pure virtual function sound
};
class Dog : public Animal {                     // Derived class Dog from Animal
public:
    void sound() const override {
        cout << "Dog: Woof!" << endl;
    }
};
class Cat : public Animal {                      // Derived class Cat from Animal
public:
    void sound() const override {
        cout << "Cat: Meow!" << endl;
    }
};
class Cow : public Animal {                      // Derived class Cow from Animal
public:
    void sound() const override {
        cout << "Cow: Moo!" << endl;
    }
};
int main() {
    Animal* animals[3];                         // Declare an array of Animal pointers
    Dog dog;                                        // Instantiate objects of Dog, Cat, and Cow
    Cat cat;
```

```
Dog: Woof!
Cat: Meow!
Cow: Moo!


...Program finished with exit code 0
Press ENTER to exit console.
```

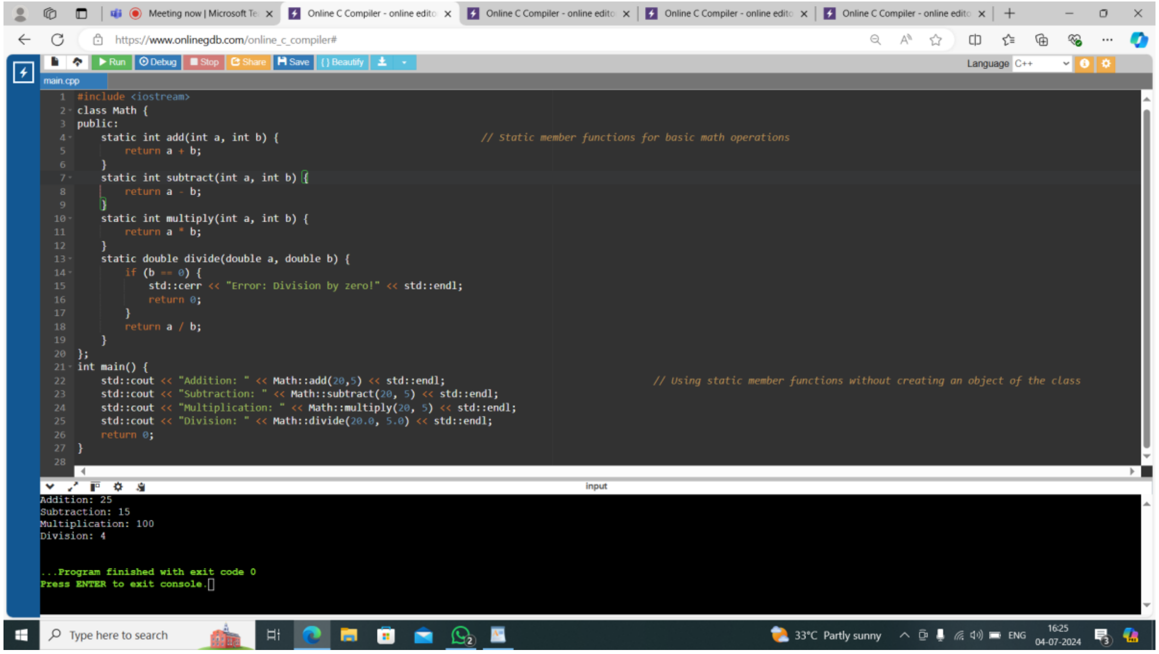
Static Member Functions:

Implement a class Math that has static member functions for basic mathematical operations like addition, subtraction, multiplication, and division. Demonstrate the use of these functions without creating an object of the class.

```
#include <iostream>

class Math {

public:

        static int add(int a, int b) {                                                            // Static member functions
for basic math operations

                return a + b;

        }

        static int subtract(int a, int b) {

                return a - b;
```

```
    }
    static int multiply(int a, int b) {
        return a * b;
    }
    static double divide(double a, double b) {
        if (b == 0) {
            std::cerr << "Error: Division by zero!" << std::endl;
            return 0;
        }
        return a / b;
    }
};
int main() {
    std::cout << "Addition: " << Math::add(20,5) << std::endl;
// Using static member functions without creating an object of the class
    std::cout << "Subtraction: " << Math::subtract(20, 5) << std::endl;
    std::cout << "Multiplication: " << Math::multiply(20, 5) << std::endl;
    std::cout << "Division: " << Math::divide(20.0, 5.0) << std::endl;
    return 0;
}
```

OUTPUT :-



Meeting now | Microsoft Te
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
https://www.onlinegdb.com/online_c_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
input
Type here to search
33°C Partly sunny
ENG
16:25
04-07-2024

```cpp
#include <iostream>
class Math {
public:
    static int add(int a, int b) {                              // Static member functions for basic math operations
        return a + b;
    }
    static int subtract(int a, int b) {
        return a - b;
    }
    static int multiply(int a, int b) {
        return a * b;
    }
    static double divide(double a, double b) {
        if (b == 0) {
            std::cerr << "Error: Division by zero!" << std::endl;
            return 0;
        }
        return a / b;
    }
};
int main() {
    std::cout << "Addition: " << Math::add(20,5) << std::endl;                      // Using static member functions without creating an object of the class
    std::cout << "Subtraction: " << Math::subtract(20, 5) << std::endl;
    std::cout << "Multiplication: " << Math::multiply(20, 5) << std::endl;
    std::cout << "Division: " << Math::divide(20.0, 5.0) << std::endl;
    return 0;
}
```

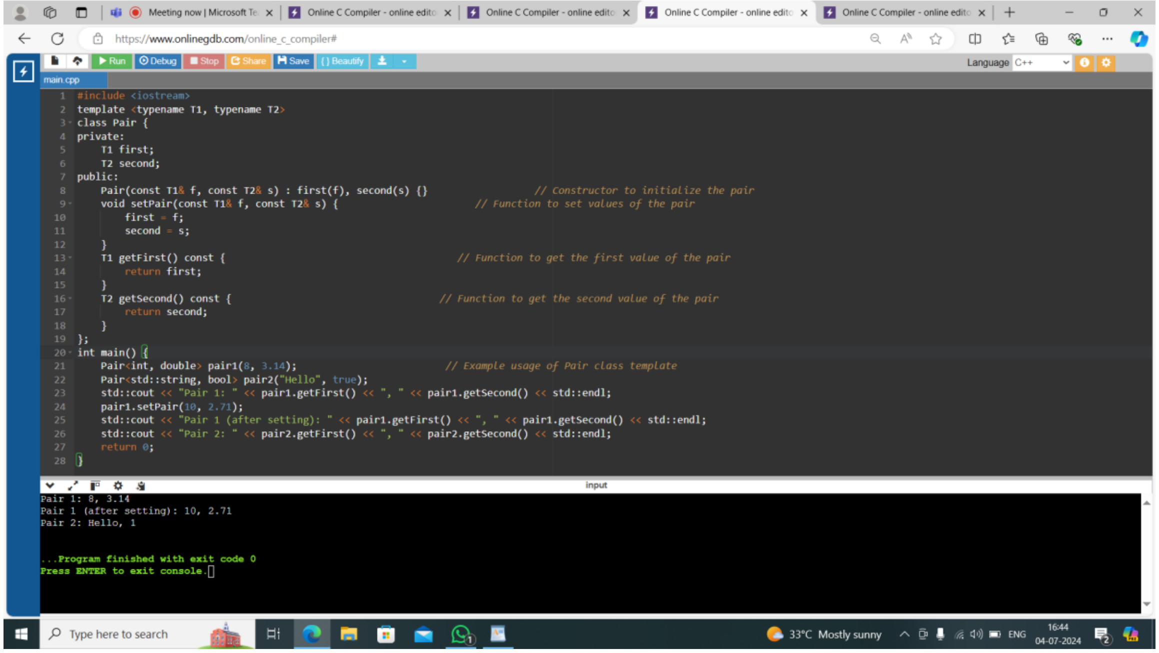
```
Addition: 25
Subtraction: 15
Multiplication: 100
Division: 4


...Program finished with exit code 0
Press ENTER to exit console.
```

Class Templates with Multiple Parameters:

Write a class template Pair that can store a pair of values of any two data types. Include member functions to set and get the values. Demonstrate the usage of this template with different data types.

```cpp
#include <iostream>

template <typename T1, typename T2>

class Pair {

private:

    T1 first;

    T2 second;

public:

    Pair(const T1& f, const T2& s) : first(f), second(s) {}                    // Constructor to initialize the pair

    void setPair(const T1& f, const T2& s) {                    // Function to set values of the pair

        first = f;

        second = s;

    }
    T1 getFirst() const {                                   // Function to get the first value of the pair
        return first;
    }
    T2 getSecond() const {                                  // Function to get the second value of the pair
        return second;
    }
};
int main() {
    Pair<int, double> pair1(8, 3.14);                       // Example usage of Pair class template
    Pair<std::string, bool> pair2("Hello", true);
    std::cout << "Pair 1: " << pair1.getFirst() << ", " << pair1.getSecond() << std::endl;
    pair1.setPair(10, 2.71);
    std::cout << "Pair 1 (after setting): " << pair1.getFirst() << ", " << pair1.getSecond() << std::endl;
    std::cout << "Pair 2: " << pair2.getFirst() << ", " << pair2.getSecond() << std::endl;
    return 0;
}
```

OUTPUT :-



Meeting now | Microsoft Te
Online C Compiler - online edito
https://www.onlinegdb.com/online_c_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
input

```cpp
#include <iostream>
template <typename T1, typename T2>
class Pair {
private:
    T1 first;
    T2 second;
public:
    Pair(const T1& f, const T2& s) : first(f), second(s) {}                // Constructor to initialize the pair
    void setPair(const T1& f, const T2& s) {                     // Function to set values of the pair
        first = f;
        second = s;
    }
    T1 getFirst() const {                                      // Function to get the first value of the pair
        return first;
    }
    T2 getSecond() const {                                   // Function to get the second value of the pair
        return second;
    }
};
int main() {
    Pair<int, double> pair1(8, 3.14);                        // Example usage of Pair class template
    Pair<std::string, bool> pair2("Hello", true);
    std::cout << "Pair 1: " << pair1.getFirst() << ", " << pair1.getSecond() << std::endl;
    pair1.setPair(10, 2.71);
    std::cout << "Pair 1 (after setting): " << pair1.getFirst() << ", " << pair1.getSecond() << std::endl;
    std::cout << "Pair 2: " << pair2.getFirst() << ", " << pair2.getSecond() << std::endl;
    return 0;
}
```

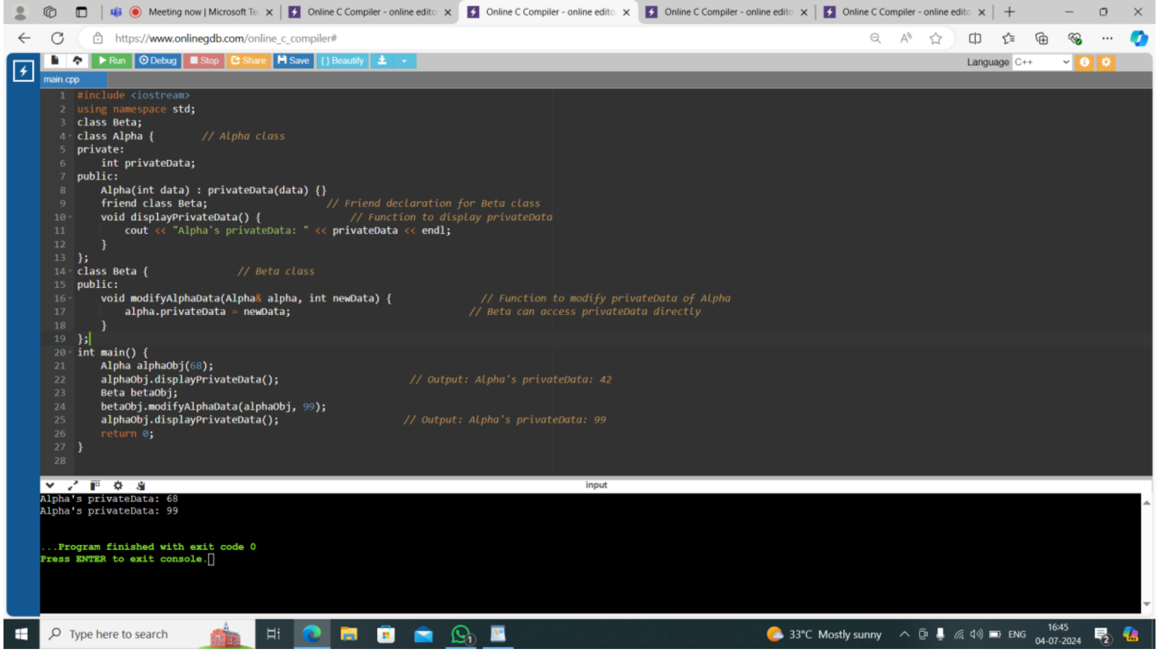
```
Pair 1: 8, 3.14
Pair 1 (after setting): 10, 2.71
Pair 2: Hello, 1


...Program finished with exit code 0
Press ENTER to exit console.
```

Friend Classes:

Create two classes Alpha and Beta. Make Beta a friend class of Alpha so that it can access private data members of Alpha. Implement functions in Beta to manipulate the private data of Alpha.

```cpp
#include <iostream>

using namespace std;

class Beta;

class Alpha {                 // Alpha class

private:

    int privateData;

public:

    Alpha(int data) : privateData(data) {}

    friend class Beta;                                    // Friend declaration for Beta class

    void displayPrivateData() {                              // Function to display privateData

        cout << "Alpha's privateData: " << privateData << endl;

    }
};
class Beta {                    // Beta class
public:
    void modifyAlphaData(Alpha& alpha, int newData) {                    // Function to modify privateData of Alpha
        alpha.privateData = newData;                                    // Beta can access privateData directly
    }
};
int main() {
    Alpha alphaObj(68);
    alphaObj.displayPrivateData();                          // Output: Alpha's privateData: 68
    Beta betaObj;
    betaObj.modifyAlphaData(alphaObj, 99);
    alphaObj.displayPrivateData();                          // Output: Alpha's privateData: 99
    return 0;
}
```

OUTPUT :-



Meeting now | Microsoft Te
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
Online C Compiler - online edito
https://www.onlinegdb.com/online_c_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
input
Type here to search
33°C Mostly sunny
ENG
16:45
04-07-2024

```
#include <iostream>
using namespace std;
class Beta;
class Alpha {        // Alpha class
private:
    int privateData;
public:
    Alpha(int data) : privateData(data) {}
    friend class Beta;                     // Friend declaration for Beta class
    void displayPrivateData() {                // Function to display privateData
        cout << "Alpha's privateData: " << privateData << endl;
    }
};
class Beta {              // Beta class
public:
    void modifyAlphaData(Alpha& alpha, int newData) {              // Function to modify privateData of Alpha
        alpha.privateData = newData;                             // Beta can access privateData directly
    }
};
int main() {
    Alpha alphaObj(68);
    alphaObj.displayPrivateData();                       // Output: Alpha's privateData: 42
    Beta betaObj;
    betaObj.modifyAlphaData(alphaObj, 99);
    alphaObj.displayPrivateData();                       // Output: Alpha's privateData: 99
    return 0;
}
```

```
Alpha's privateData: 68
Alpha's privateData: 99


...Program finished with exit code 0
Press ENTER to exit console.
```