**IK. Gujral Punjab Technical University, Kapurthala Department of Computer Science & Engineering**

****

**B. Tech.(CSE) Semester-3rd**

**(OOPS) OBJECT ORIENTED PROGRAMMING LAB LABORATORY ASSIGNMENT**

**Batch – D1**

**Submitted To: Submitted By:**

Dr. Pooja Sharma Raj yadav

(2224499)

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Question** | **Page Number** |
| 1. | Inline function  without classes | 3 |
| 2. | Inline function  without classes | 4 |
| 3. | Typecasting | 5 |
| 4. | Static data member | 6 |
| 5. | Static member  function | 7 |
| 6. | Destructor | 8 |
| 7. | Classes and objects, where member function are defined both inside and outside the class | 9 to 10 |
| 8. | Array of object for a class student and complete the result of students. take the attributes accordingly | 11 to 12 |
| 9. | Friend function Friend function | 13 to 14 |
| 10. | Constructor | 15 to 20 |

**INDEX**

1. Write a program to demonstrate the concept of :
   1. Inline function without classes

#include <iostream>

inline *int* add(*int* *a*, *int* *b*) {

    return *a* + *b*;

}

*int* main() {

*int* num1 = 5;

*int* num2 = 10;

*int* result = add(num1, num2);

    std::cout << "Result of adding " << num1 << " and " << num2 << " is: " << result << std::endl;

    return 0;

}



* 1. Function overloading

#include <iostream>

using *namespace* std;

*void* display(*int* *x*) {

    cout << "The number is " << *x* << endl;

}

*void* display(*double* *x*) {

    cout << "The number is " << *x* << endl;

}

*int* main() {

*int* a = 20;

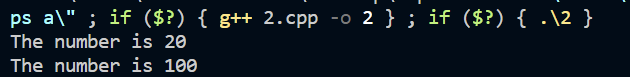
*double* b = 100.0;

    display(a);

    display(b);

    return 0;

}



* 1. Typecasting

#include <iostream>

using *namespace* std;

*int* main() {

*int* i = 45;

*float* f = 3.14;

*int* j = (*int*)f;

*float* k = (*float*)i;

*float* l = i + f;

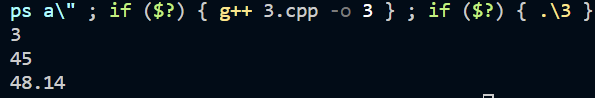
    cout << j << endl;

    cout << k << endl;

    cout << l << endl;

    return 0;

}



* 1. Static data member

#include <iostream>

using *namespace* std;

*class* A {

*public:*

    A() {

        cout << "A's Constructor Called " << endl;

    }

};

*class* B {

    static A a;

*public:*

    B() {

        cout << "B's Constructor Called " << endl;

    }

};

*int* main() {

    B b;

    return 0;

}



1. Static member function

#include <iostream>

using *namespace* std;

*class* Test {

*public:*

    static *int* count;

    static *void* increment() {

        count++;

    }

    static *int* getcount() {

        return count;

    }

};

*int* Test::count = 0;

*int* main() {

    Test t1, t2, t3;

    t1.increment();

    t2.increment();

    t3.increment();

cout << "The count is: " << Test::getcount() << endl;

return 0;

}



1. Destructor

#include <iostream>

using *namespace* std;

*class* Test {

*public:*

    Test() {

        cout << "\nConstructor executed";

    }

    ~Test() {

        cout << "\nDestructor executed";

    }

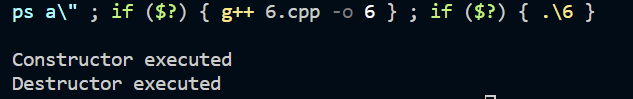
};

*int* main() {

    Test t;

    return 0;

}



1. Classes and objects, where member function are defined both inside and outside the class.

#include <iostream>

using *namespace* std;

*class* MyClass {

*public:*

*int* x;

*int* y;

    MyClass(*int* *a*, *int* *b*);

*void* display();

*void* addValues();

};

MyClass::MyClass(*int* *a*, *int* *b*) {

    x = *a*;

    y = *b*;

}

*void* MyClass::display() {

    cout << "x = " << x << ", y = " << y << endl;

}

*void* MyClass::addValues() {

*int* sum = x + y;

    cout << "Sum of x and y is: " << sum << endl;

}

*int* main() {

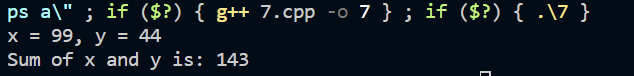
    MyClass obj(99, 44);

    obj.display();

    obj.addValues();

    return 0;

}



1. Array of object for a class student and complete the result of students. take the attributes accordingly.

#include <iostream>

#include <string>

using *namespace* std;

*class* Student {

*public:*

    string name;

*int* rollNumber;

*double* marks;

    Student(string *n*, *int* *rn*, *double* *m*) {

        name = *n*;

        rollNumber = *rn*;

        marks = *m*;

    }

*void* display() {

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

        cout << "Roll Number: " << rollNumber << endl;

        cout << "Marks: " << marks << endl;

    }

};

*int* main() {

    const *int* numStudents = 3;

    Student students[numStudents] = {

        Student("Ansh",234, 92.5),

        Student("Joe", 235, 98.0),

        Student("Ash", 236, 88.5)

    };

    for (*int* i = 0; i < numStudents; i++) {

        cout << "Student " << (i + 1) << ":\n";

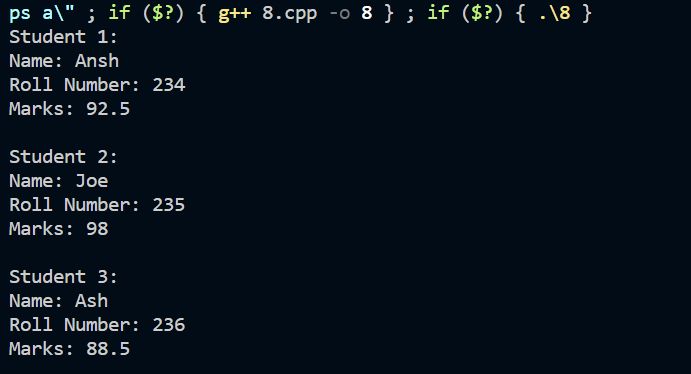
        students[i].display();

        cout << endl;

    }

    return 0;

}



1. Friend function
   1. Without classes

#include <iostream>

*void* friendFunction(*int* *num*);

*void* regularFunction(*int* *num*) {

    std::cout << "Inside regularFunction. Received number: " << *num* << std::endl;

}

*int* main() {

*int* number = 82;

    regularFunction(number);

    friendFunction(number);

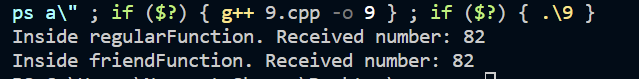
    return 0;

}

*void* friendFunction(*int* *num*) {

    std::cout << "Inside friendFunction. Received number: " << *num* << std::endl;

}



* 1. With Classes

#include <iostream>

*class* MyClass {

*private:*

*int* privateData;

*public:*

    MyClass(*int* *data*) : privateData(*data*) {}

    friend *void* showPrivateData(const MyClass& *obj*);

};

*void* showPrivateData(const MyClass& *obj*) {

    std::cout << "Private Data: " << *obj*.privateData << std::endl;

}

*int* main() {

    MyClass obj(67);

    showPrivateData(obj);

    return 0;

}



1. Constructor
   1. Default

#include <iostream>

*class* MyClass {

*public:*

    MyClass() {

        std::cout << "Default constructor called" << std::endl;

    }

*void* displayMessage() {

        std::cout << "Hello Everyone!" << std::endl;

    }

};

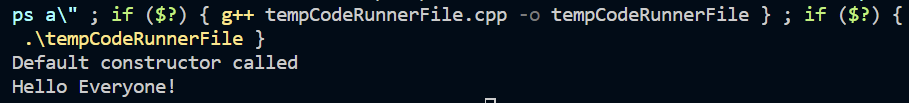
*int* main() {

    MyClass obj;

    obj.displayMessage();

    return 0;

}



* 1. Parameterized

#include <iostream>

*class* MyClass {

*public:*

    MyClass(*int* *value*) {

        this->value = *value*;

    }

*void* displayValue() {

        std::cout << "Value: " << value << std::endl;

    }

*private:*

*int* value;

};

*int* main() {

    MyClass obj1(45);

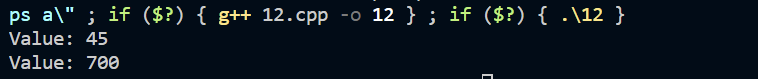
    obj1.displayValue();

    MyClass obj2(700);

    obj2.displayValue();

    return 0;

}



* 1. Copy

#include <iostream>

*class* MyClass {

*public:*

    MyClass(*int* *value*) : data(*value*) {

        std::cout << "Constructor called" << std::endl;

    }

    MyClass(const MyClass& *other*) : data(*other*.data) {

        std::cout << "Copy constructor called" << std::endl;

    }

*void* display() {

        std::cout << "Data: " << data << std::endl;

    }

*private:*

*int* data;

};

*int* main() {

    MyClass obj1(55);

    MyClass obj2(obj1);

    std::cout << "Object 1: ";

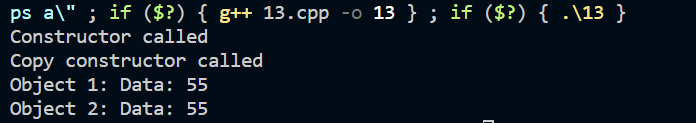
    obj1.display();

    std::cout << "Object 2: ";

    obj2.display();

    return 0;

}



d. Constructor overloading

#include <iostream>

using *namespace* std;

*class* MyClass {

*private:*

*int* num1;

*int* num2;

*public:*

    MyClass() {

        num1 = 0;

        num2 = 0;

    }

    MyClass(*int* *n*) {

        num1 = *n*;

        num2 = 0;

    }

    MyClass(*int* *n1*, *int* *n2*) {

        num1 = *n1*;

        num2 = *n2*;

    }

*void* display() {

        cout << "num1: " << num1 << ", num2: " << num2 << endl;

    }

};

*int* main() {

    MyClass obj1;

    MyClass obj2(29);

    MyClass obj3(90, 40);

    cout << "Object 1: ";

    obj1.display();

    cout << "Object 2: ";

    obj2.display();

    cout << "Object 3: ";

    obj3.display();

    return 0;

}

