This article is mainly focused on the most repeatedly asked and the latest updated interview questions on Inheritance in C++ that are appearing in most of the C++ interviews.

If you are looking for "C++ interview questions on Inheritance " or "advanced questions on Inheritance in C++, then you at the right place. Here I have tried to create some collection of "Some interview questions with answers related to the Inheritance in C+ " that might ask by your interviewer.

I hope these **C++ interview questions (https://aticleworld.com/cpp-interview-questions/)** with the answer will be helpful. If you have any other important questions on Inheritance in C++ programming and concept, then please write in the comment box. It will be helpful for others.

Q #1) What Is Inheritance?

Inheritance allows us to define a class that inherits all the methods and attributes from another class. The class that inherits from another class is called a derived class or child class. The class from which we are inheriting is called parent class or base class.

Q #2)What are a Base and derived class?

In inheritance, the existing class is called the base or parent class and the newly created class is called derived or child class. A derived class can be inherited from more than one class, it all the thing depends on the requirements. When we have created a derived class then derived class able to reuse the code of the base class.

Q #3) How to implement inheritance?

For creating a sub-class that is inherited from the base class we have to follow the below syntax.

```
class derived_class : access_specifier base_class
{
  //body of derived_class
};
```

The syntax of inheritance in C++ is very simple. You just create a class as usual but before the opening of braces of the body of class just put a colon and name of the base class with the access specifier.

Here access specifier can be public, private or protected, the derived class is newly created class and base class is already existing class.

Q #4) What are C++ access modifiers?

There are 3 types of access modifiers available in C++:

Public: There are no restrictions on accessing public members. The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

Private: Access is limited to within the class definition. This is the default access modifier type if none is formally specified. They are not allowed to be accessed directly by any object or function outside the class.

Protected: Access is limited to within the class definition and any class that inherits from the class.

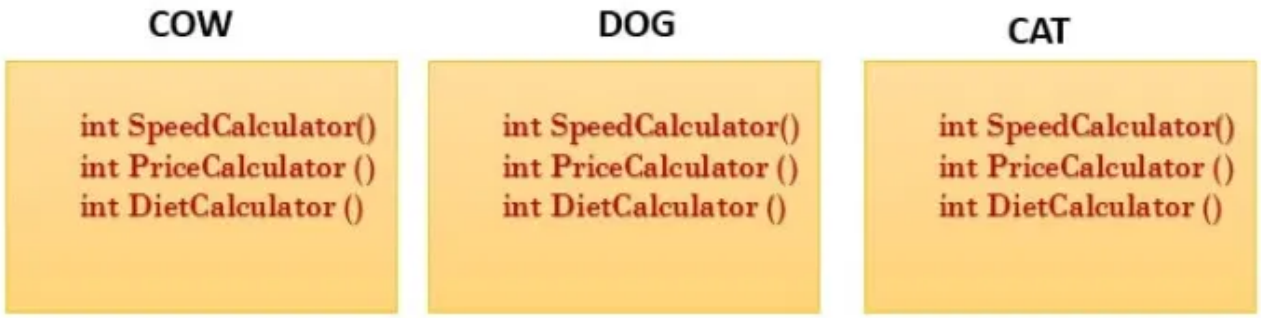
Q #5) Why use access modifiers in C++?

Access modifiers are an integral part of object-oriented programming. They are used to implement the encapsulation of OOP. The access modifiers allow you to define who does or who doesn't have access to certain features.

Q #6) Why is inheritance required?

Suppose in a program you have required to collect the information of cow, dog, and cat. This information is like their speed, price, and diet. So you have to create three-class here to save the information of cow, dog and cat and each class contains the function to calculate their speed price and diet.

```
//class which contains the information of Cow
class Cow
{
public:
    int SpeedCalculator;
    int PriceCalculator;
    int DietCalculator;
};
//class which contains the information of Dog
class Dog
{
public:
    int SpeedCalculator;
    int PriceCalculator;
    int DietCalculator;
};
//class which contains the information of Cat
class Cat
{
public:
    int SpeedCalculator;
    int PriceCalculator;
    int DietCalculator;
};
```



From the above image, it is clear that these three classes use the same three functions to calculate the speed, price, and diet. This type of approach is not good for development and it reduces the code reusability and increases the development time.

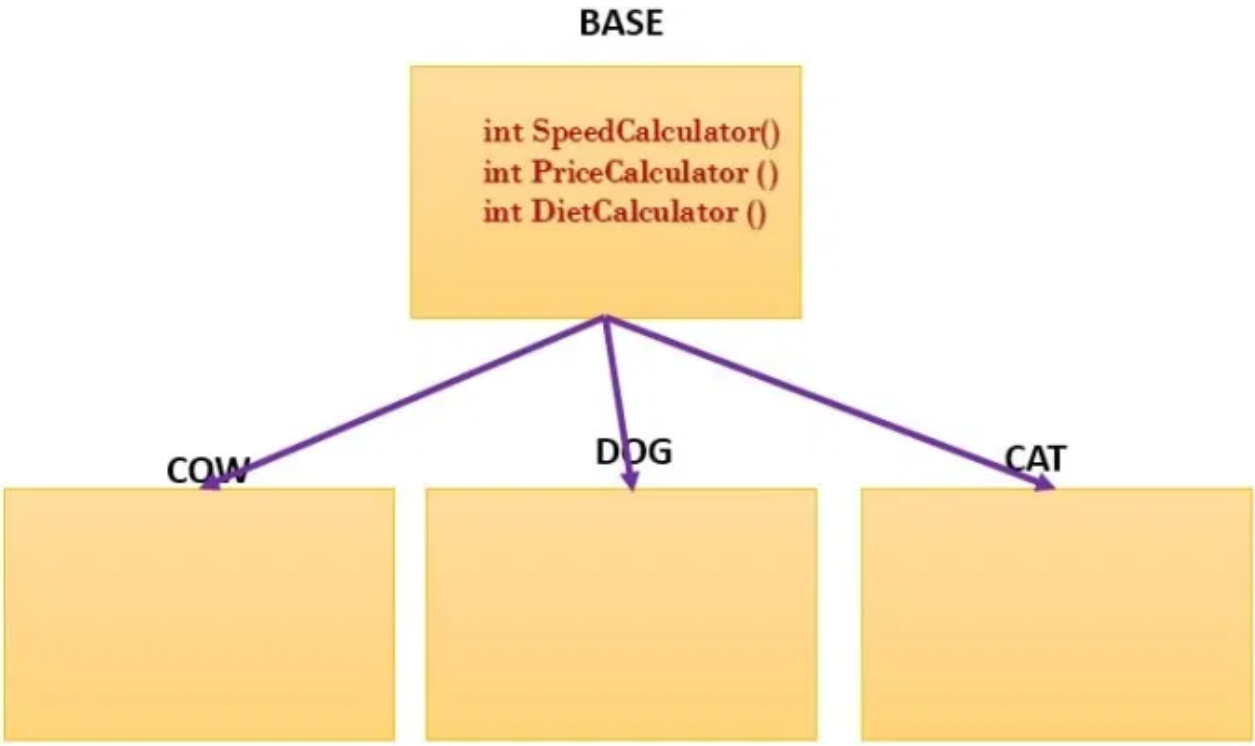
We can solve this problem with the help of inheritance. We just need to create a base class that will consist of all the three-member function and just inherits this base class for the class which is used to store the information of the cow, dog, and cat.

```
//Animal is base class
class Animal
{
public:
    int SpeedCalculator();
    int PriceCalculator();
    int DietCalculator();
};

//Cow class is child class of the animal class
class Cow : public Animal
{
}

//Dog class is child class of the animal class
class Dog : public Animal
{
}

//Cat class is child class of the animal class
class Cat : public Animal
{
}
```



Q #7) What are the advantages of inheritance?

There are many benefits of inheritance in C++, so let us see them:

- Inheritance provides code reusability, makes it easier to create and maintain an application. So we don't have to write the same code again and again.
- It allows us to add more features to a class without modifying it.
- It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.
- Inheritance represents real-world relationships well.

Q #8) More than one class may be derived from a base class

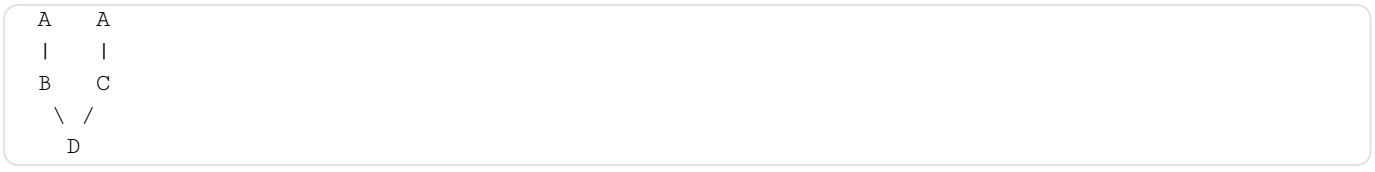
Yes.

Q #9) What are the types of inheritance?

Single inheritance
Multiple inheritances.
Multilevel inheritance
Hierarchical inheritance
Hybrid inheritance

Q #10) What is the Diamond problem? How can we get around it?

C++ allows multiple inheritances. Multiple inheritances allow a child class to inherit from more than one parent class. The diamond problem occurs when two superclasses of a class have a common base class. For example, in the following diagram, the “D class” gets two copies of all attributes of “A class”, this causes ambiguities. Let see the below image which shows what happens without virtual inheritance?



The solution to this problem is the ‘virtual’ keyword. We make the classes “B” and “C” as virtual base classes to avoid two copies of class “A” in the “D” class.

```
B   C
 \ /
  D
```

Now times to see some programming Questions Related to inheritance:

Q #11) What is the output of the below program?

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


class BaseA
{
public:
    BaseA()
    {
        cout << "BaseA constructor called" << endl;
    }
};

class BaseB
{
public:
    BaseB()
    {
        cout << "BaseB constructor called" << endl;
    }
};

class Derived: public BaseA, public BaseB
{
public:
    Derived()
    {
        cout << "Derived's constructor called" << endl;
    }
};

int main()
{
    Derived d;
    return 0;
}
```

Output:

BaseA constructor called
BaseB constructor called
Derived's constructor called

Explanation: When a class inherits from multiple classes, constructors of base classes are called in the same order as they are specified in inheritance.

Q #12) What is the output of the below program?

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

class Base1
{
public:
    char data;
};

class Base2
{
public:
    int data;
};

class Child: public Base1, public Base2
{
public:
    void show()
    {
        cout << Base2::data;
    }
};

int main(void)
{
    Child d;
    d.show();
    return 0;
}
```

Output:

Garbage.

Explanation: The variable "data" is present in both base classes. So the access to "data" in the child class is ambiguous. The ambiguity can be removed by using the scope resolution operator.

Q #13) What is the output of the below program? Assumed size of int is 4 byte.

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

class Base
{
    int arr[5];
};

class Child1: public Base
{

};

class Child2: public Base
{

};

class GrandChild: public Child1, public Child2
{

};

int main(void)
{
    cout << sizeof(GrandChild);
    return 0;
}
```

Output:

40 bytes

Explanation: Since Child1 and Child12 both inherit from the base class, two copies of the class base are there in class Grandchild. It is called diamond problem.

If you want to learn the C++11 from scratch then you can follow this course (https://pluralsight.pxf.io/c/1192901/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fcplusplus-11-from-scratch) trial is free.

Your free trial is waiting. (https://pluralsight.pxf.io/c/1192901/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fcplusplus-11-from-scratch)

Q #14) What is the output of the below program?

```
#include<iostream>

using namespace std;

class A
{
public:
    void display()
    {
        cout <<" Inside A";
    }
};

class B : public A
{
public:
    void display()
    {
        cout <<" Inside B";
    }
};

class C: public B
{

};

int main(void)
{
    C C;
    C.display();
    return 0;
```

```
}
```

Output:

Inside B

Explanation: The display function is not present in class “C”. So it is looked up in the inheritance hierarchy. The display() is present in both classes “A” and “B”, which of them should be called? The idea is, if there is multilevel inheritance, then the function is linearly searched up in the inheritance hierarchy until a matching function is found.

Q #15) What is the output of the below program?

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

class Base
{
private:
    int data1, data2;

public:
    Base(int a = 0, int b = 0): data1(a), data2(b)
    {

    }
};


class Derived: public Base
{
public:
    void show()
    {
        cout<<" data1 = "<<data1<<" data2 = "<<data2;
    }
};


int main(void)
{
    Derived d;
    d.show();
    return 0;
}
```

Output:

compiler error.

Explanation: Try to access private members of the base class.

Q #16) What is the output of the below program?

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

class Base
{
public:
    int data1, data2;

public:
    Base(int a = 0, int b = 0): data1(a), data2(b)
    {

    }
};


class Derived: public Base
{
public:
    void show()
    {
        cout<<" data1 = "<<data1<<" data2 = "<<data2;
    }
};


int main(void)
{
    Derived d;
    d.show();
    return 0;
}
```

Output:

data1 = 0 data2 = 0

Q #17) What is the output of the below program?

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

class Base
{
```

```
};

class Derived: public Base
{

};

int main()
{
    Base *bp = new Derived;
    Derived *dp = new Base;
}
```

Output:

compiler error.

Explanation: A Base class pointer/reference can point/refer to a derived class object, but the other way is not possible.

Q #18) What is the output of the below program?

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

class Base
{
public:
    void print()
    {
        cout<<" In Base ";
    }
};

class Child: public Base
{
public:
    int data1;
    void print()
    {
        cout<<"In Child ";
    }
    Child()
    {
        data1 = 10;
    }
};

int main(void)
{
    Base *bp;

    Child d;
```

```
    bp = &d;

    bp->print();

    cout << bp->data1;

    return 0;
}
```

Output:

compiler error.

Explanation: A base class pointer can point to a derived class object, but we can only access base class members or virtual functions using the base class pointer because object slicing happens.

When a derived class object is assigned to a base class object. Additional attributes of a derived class object are sliced off to form the base class object.

Q #19) What is the output of the below program?

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

class Base
{
public:
    virtual string print() const
    {
        return "This is Base class";
    }
};

class Child : public Base
{
public:
    virtual string print() const
    {
        return "This is Child class";
    }
};

void describe(Base p)
{
    cout << p.print() << endl;
}
```

```
int main()
{
    Base b;
    Child d;
    describe(b);
    describe(d);

    return 0;
}
```

Output:

This is Base class
This is Base class

Explanation: When we assign an object of the derived class to an object of the base type, the derived class object is sliced off and all the data members inherited from the base class are copied.

Q #20) What is the output of the below program?

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

class Base
{
public:
    virtual string print() const
    {
        return "This is Base class";
    }
};

class Child : public Base
{
public:
    virtual string print() const
    {
        return "This is Child class";
    }
};

void describe(Base *p)
{
    cout << p->print() << endl;
}

int main()
{
    Base b;
```

```
    Child d;
    describe(&b);
    describe(&d);
    return 0;
}
```

Output:

This is Base class

This is Child class

Q #21) What is the output of the below program?

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


class A
{
public:
    A()
    {
        cout <<"constructor A"<<endl;
    }
    A(const A &obj)
    {
        cout <<"copy constructor A"<<endl;
    }
};

class B: virtual A
{
public:
    B()
    {
        cout <<"constructor B"<<endl;
    }
    B(const B & obj)
    {
        cout<<"copy constructor B"<<endl;
    }
};

class C: virtual A
{
public:
    C()
    {
        cout<<"constructor C"<<endl;
    }
    C(const C & obj)
    {
        cout <<"copy constructor C" <<endl;
    }
```

```
};

class D:B,C
{
public:
    D()
    {
        cout<<"constructor D" <<endl;
    }
    D(const D & obj)
    {
        cout <<"copy constructor D" <<endl;
    }
};

int main()
{
    D obj1;

    cout  <<endl <<"Now Creating Obj2" <<endl;
    D obj2(obj1);
}
```

Output:

```
constructor A
constructor B
constructor C
constructor D

Now Creating Obj2
constructor A
constructor B
constructor C
copy constructor D
```

Explanation: In inheritance, we need to explicitly call the copy constructor of base class otherwise only default constructor of the base class is called.

Q #22) What is the output of the below program?

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

class Base
{
public :
  int data1, data2;
public:
  Base(int i, int j){ data1 = i; data2 = j; }
```

```
};

class Child : public Base
{
public:
  Child(int i, int j):data1(i), data2(j) {}
  void print() {cout << data1 <<" "<< data2; }
};

int main(void)
{
  Child q(10, 10);
  q.print();
  return 0;
}
```

Output:

Compiler Error

Explanation: The base class members cannot be directly assigned using the initializer list. We should call the base class constructor to initialize base class members.

Q #23) What is the output of the below program?

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

class Base
{
public :
    int data1, data2;
public:
    Base(int i, int j)
    {
        data1 = i;
        data2 = j;
    }
};


class Child : public Base
{
public:
    Child(int i, int j):Base(i, j)
    {
    }
    void print()
    {
        cout << data1 <<" "<< data2;
    }
};
```

```
int main(void)
{
    Child q(9, 9);
    q.print();
    return 0;
}
```

Output:

9 9

Q #24) What is the output of the below program?

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

class Base
{
public:
    void fun()
    {
        cout << "Base::fun() called";
    }
    void fun(int i)
    {

        cout << "Base::fun(int i) called";
    }
};

//child class
class Child: public Base
{
public:
    void fun()
    {
        cout << "Child::fun() called";
    }
};

int main()
{
    Child d;
    d.fun(27);
    return 0;
}
```

Output:

Compiler Error

Explanation: In the above program, when fun() is rewritten in the child, it hides both fun() and fun(int) of the base class.

When a child class writes its own method, then all functions of the base class with the same name become hidden, even if signatures of base class functions are different.

Q #25) What is the output of the below program?

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

class Base
{
protected:
    int a;
public:
    Base()
    {
        a = 0;
    }
};

class Child1: public Base
{
public:
    int c;
};


class Child2: public Base
{
public:
    int c;
};

class GrandChild: public Child1, public Child2
{
public:
    void show()
    {
        cout << a;
    }
};

int main(void)
{
    GrandChild d;
    d.show();
    return 0;
}
```

Output:

Compiler Error

Explanation: Here the base class member “a” is inherited through both Child1 and Child12. So there are two copies of ‘a’ in GrandChild which makes “a” ambiguous.

Q #26) What is the output of the below program?

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

class Base
{
protected:
    int a;
public:
    Base()
    {
        a = 10;
    }
};

class Child1: virtual public Base
{
public:
    int c;
};


class Child2: virtual public Base
{
public:
    int c;
};

class GrandChild: public Child1, public Child2
{
public:
    void show()
    {
        cout << a;
    }
};

int main(void)
{
    GrandChild d;
    d.show();
    return 0;
}
```

Output:

Explanation: Using the virtual keyword we can resolve the diamond problem.

Q #27)Do all virtual functions need to be implemented in derived classes?

The derived classes do not have to implement all virtual functions themselves. See the below example code,

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

class base
{
public:
    virtual void print()
    {
        cout << "print base class" << endl;
    }

    virtual void display()
    {
        cout << "print base class" << endl;
    }


};


class derived: public base
{
public:
    void print()
    {
```

```
        cout << "print derived class" << endl;
    }
};


int main(void)
{
    //derive class object
    derived d;
    //Base class pointer
    base *b = &d;

    // virtual function, binded at runtime
    b->print();

    return 0;
}
```

Output:

print derived class

Q #28)Do all pure virtual functions need to be implemented in derived classes?

We have to implement all pure virtual functions in derived class only if the derived class is going to be instantiated. But if the derived class becomes a base class of another derived class and only exists as a base class of more derived classes, then derived class responsibility to implement all their pure virtual functions.

The “middle” class in the hierarchy is allowed to leave the implementation of some pure virtual functions, just like the base class. If the “middle” class does implement a pure virtual function, then its descendants will inherit that implementation, so they don’t have to re-implement it themselves. Let see an example code to understand the concept.

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

class ISuperbase
{
public:
    virtual void print() = 0;
    virtual void display() = 0;
};


class Base: public ISuperbase
{
public:
```

```
    virtual void print()
    {
        cout << "print function of middle class" << endl;
    }
};


class Derived :public Base
{
    virtual void display()
    {
        cout << "In display function" << endl;
    }
};


int main(void)
{
    //derive class object
    Derived d;

    // virtual function, binded at runtime
    d.print();

    return 0;
}
```

Output:

print function of middle class

Q #29) How to call a parent class function from a derived class function?

If a function defined in a base class and it is not a private then it is available in the derived class. You can call it in the derived class using the resolution operator (::). Let see a code where I am accessing parent class function in the derived class as well as from the derived class object.

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



class Base
{
public:
    virtual void print()
    {
        cout << "I am from base class" << endl;
    }
};
```

```
class Derived :public Base
{
    void display()
    {
        //calling base class function
        Base::print();
    }
};


int main()
{
    //derive class object
    Derived d;

    //calling print function
    d.print();

    //Calling print function of parent class
    // using derived class object
    d.Base::print();

    return 0;
}
```

Output:

I am from base class

I am from base class

Recommended Articles for you:

- C++ Interview Questions with Answers. (https://aticleworld.com/cpp-interview-questions/)
- How to create dynamic array in C? (https://aticleworld.com/dynamically-allocate-2d-array-c/)
- Memory Layout in C. (https://aticleworld.com/memory-layout-of-c-program/)
- 100 embedded C interview Questions. (https://aticleworld.com/embedded-c-interview-questions-2/)
- Python Interview Questions with Answer. (https://aticleworld.com/python-interview-questions/)
- 100 c interview questions, your interviewer might ask. (https://aticleworld.com/c-interview-questions/)

- **C Interview Questions for the experience. (https://aticleworld.com/c-interview-questions-for-experienced-with-answer/)**
- **File handling in C. (https://aticleworld.com/file-handling-in-c/)**
- **C-Sharp Interview Questions. (https://aticleworld.com/c-sharp-interview-questions/)**

About (https://aticleworld.com/author/pritosh/)

I am an embedded c software engineer and a corporate trainer, currently, I am working as senior software engineer in a largest Software consulting company . I have working experience of different microcontrollers (stm32, LPC, PIC AVR and 8051), drivers (USB and virtual com-port), POS device (VeriFone) and payment gateway (global and first data).

WEBSITE (HTTPS://ATICLEWORLD.COM/) TWITTER (HTTPS://TWITTER.COM/HTTPS://TWITTER.COM/ATICLEWORLD) FACEBOOK (HTTPS://WWW.FACEBOOK.COM/ATICLEWORLD-602608163238897/) LINKEDIN (HTTPS://WWW.LINKEDIN.COM/IN/AMLENDRA-KUMAR-BB3B2096/) INSTAGRAM (HTTPS://WWW.INSTAGRAM.COM/ATICLEWORLD/)

← PREVIOUS ARTICLE (HTTPS://ATICLEWORLD.COM/INTERVIEW-QUESTIONS-ON-VIRTUAL-KEYWORD-IN-C/)

NEXT ARTICLE (HTTPS://ATICLEWORLD.COM/HOW-TO-PLAY-NO-INTERNET-DINOSAUR-GAME-BOTH-ONLINE-AND-OFFLINE/) →

(https://aticleworld.com/interview-questions-on-virtual-keyword-in-c/)

Interview questions on virtual keyword in C++ (https://aticleworld.com/interview-questions-on-virtual-keyword-in-c/)

(https://aticleworld.com/how-to-play-no-internet-dinosaur-game-both-online-and-offline/)

How to Play No internet dinosaur Game – Both Online and Offline (https://aticleworld.com/how-to-play-no-internet-dinosaur-game-both-online-and-offline/)

(HTTPS://ATICLEWORLD.COM/AUTHOR/PRITOSH/)/ (HTTPS://ATICLEWORLD.COM/QUESTIONS-ON-INHERITANCE-IN-C/)

Leave a Reply

YOUR EMAIL ADDRESS WILL NOT BE PUBLISHED. REQUIRED FIELDS ARE MARKED *

COMMENT *

NAME *

EMAIL *

WEBSITE

SAVE MY NAME, EMAIL, AND WEBSITE IN THIS BROWSER FOR THE NEXT TIME I COMMENT.

POST COMMENT

(https://aticleworld.com/best-c-programming-books/)

(https://aticleworld.com/best-gift-programmers/)

Join Aticleworld

You will also get our free C interview questions eBook

Enter your email here*

Subscribe

Pages

About (https://aticleworld.com/about/)
Guest Article (https://aticleworld.com/guest-article/)
Blog Posts (https://aticleworld.com/blog-post/)
affiliate-disclosure (https://aticleworld.com/affiliate-disclosure/)
disclaimer (https://aticleworld.com/disclaimer/)

BACK TO TOP ↑