Programming blockchain part 1: C ++ programming language

Extremely attractive blockchain technology. It won't be too far away to think about a future built entirely on this technology.

Extremely attractive blockchain technology. It won't be too far away to think about a future built entirely on this technology. So do you ever wonder what you need to learn to start growing on the blockchain? Which programming language will give you an advantage? The following tutorial will detail some of the main programming languages, to provide the most comprehensive view of the blockchain for readers.

1. Here are the 3 most readable blockchain books
2. This is why 10 years from now, every company will use blockchain

Problems with blockchain software development

Before starting, let's take a look at some of the challenges that a blockchain developer faces. Creating and maintaining a public blockchain is not an easy thing for a number of reasons.

Reason # 1: Security

Blockchain is like a fortress. First, blockchain's source code is public and open to everyone. Anyone can see the code and check for errors and security holes. However, unlike other open source resources, if the vulnerability on the blockchain is found, the risk is very terrible. Any developer can hack in and "go" with millions of millions of dollars. Due to security concerns, development on blockchain is often very slow.

Reason # 2: Resource management

It is important to keep up with the development of networks. You don't lag too far behind and don't keep up with the requirements of the network. You need to be well equipped to handle local and remote queries.

Reason number 3: Performance

Blockchain must always operate with the highest possible performance, so the programming language must also be very flexible. The problem is that there are certain tasks in the blockchain that can be executed in parallel while some other tasks do not.

An example of a 'parallel' task is digital signature verification. All you need to verify the signature is the key, transaction and signature. With only the above three data, you can proceed to verify in parallel.

However, not all functions on a blockchain can do it that way. For example, executing transactions. Many transactions cannot be performed in parallel. Only one transaction is executed at a time. Just like some programming languages ??work in parallel quite well while others do not.

Reason # 4: Independence

Identifying behavior - What is Deterministic behavior?

If **A + B = C** , regardless of the circumstances, **A + B** will always be equal to **C.** That is called definite behavior. The hash function specifies, meaning that the hash value of A will always be **H (A)** .

Therefore, in the blockchain development, all transaction activities must be identified. You cannot have a transaction that works this way the day before and then works the other way the next day. Similarly, you cannot have smart contracts that operate in two different ways on two different machines.

The only solution to this is independence. Basically, you isolate your smart contracts and transactions from unknown factors.

Those are the main issues that blockchain developers face. Now, let's look at some of the languages ??developers can use to **program blockchain: C ++** , **JavaSript** , **Python, Java** and **Solidity** . In the first section, we will learn about C ++, the native bitcoin language chosen to create Bitcoin's source code.

C ++ programming language

First and foremost, start with C ++. C ++ is created by Bjarne Stroustrup as an extension of the C language. This language is designed to have the flexibility and effectiveness of C programming language but there are some big differences. The biggest difference between C and C ++ is that C is a procedural programming language, while C ++ is an object-oriented programming language.

This means that, in C ++, data and functions are grouped, called 'objects' - objects. This means that when an object has been created, it can easily be called and reused in other programs, saving a lot of time writing code.

Consider the simplest C ++ program in the world: **Hello World**

```
#include
main ()
{
cout  "Hello World!";
return 0;
}
```

So why do people still use C ++ to write code? Certainly now there are many better programming languages, why do people still insist on using C ++? Why is the bitcoin blockchain encrypted on C ++?

Because C ++ has some very interesting features as follows:

Feature 1: Memory control

Recall what you said earlier about the challenges of blockchain developers. Blockchain not only protects systems but also manages resources effectively. A blockchain will have to interact with many unreliable endpoints while still providing fast service for all nodes.

Rapid service is a crucial factor for the success of an electronic currency, like bitcoin. Remember, all based on the "consensus" principle, all nodes on the network must accept and reject exactly the same blocks, or another block can act as an intermediary in the chain.

To meet all of these requirements and perform at the highest level, you need to strictly control CPU and memory usage. Thankfully, C ++ provides users with this useful feature.

Feature 2: Stream

As we discussed earlier, one of the main challenges of blockchain programming is the good integration of parallel tasks and non-parallel tasks. Most languages ??specialize in one aspect, but C ++'s streaming capabilities are good to handle both parallel and non-parallel tasks. 'Thread' - a thread is a set of commands that can be executed simultaneously. C ++ not only allows multi-threaded utilities to flexibly operate, increase the efficiency of inter-thread communication and optimize one-stream performance.

Feature 3: Move Semantics

One of the most interesting aspects of C ++ is Move Semantics. Move Semantics provides a way for content to be moved flexibly between objects instead of being completely copied. Check the difference between Copy semantics and Move Semantics.

Copy semantics:

1. assert (b == c);
2. a = b;
3. assert (a == b && b == c);

The value of b is replaced by a and finally the value of b remains unchanged.

Now, consider this.

Move Semantics:

1. assert (b = = c);
2. move (a, b);
3. assert (a = = c);

Do you see the difference between the two code blocks?

When using Move Semantics, the value of 'b' does not need to remain the same. That's the difference between Copy semantics and Move Semantics. The biggest advantage of Move Semantics is that you can get copies of certain data only when you need them, greatly reducing redundancy in the code and delivering great performance. So, as you can see, good memory management and high performance are quite effective when applying C ++ to blockchain.

Feature 4: Gather time polymorphism

What is polymorphism?

Remember when we call C ++ an object-oriented programming language ('OOP) language'. Polymorphism - Polymorphism - is also an OOP type. Using Polymorphism means you use a specific feature in many ways. In C ++ Polymorphism can be used in two ways:

1. Polymorphism at compile time
2. Polymorphic at run time

Here, the article will focus on polymorphism at compile time. There are two ways that C ++ performs polymorphism at compile time:

1. Load Overlay (Function Overloading)
2. Operator overload (Operator Overloading)

Overlay function:

Load overlap is when you have multiple functions with the same name but with different parameters.

Review this program:

```
  #include
  using namespace std;
  class A
  {
  void func (int x) // first instance of function không ch? ch? m?t s?
 nguyên
  {
  cout    }
  void func (double x) // second instance c?
a function takes only one double value
  {
  cout    }
  void func (int x, y int) // third instance of the function takes two integer va
  {
  cout    }
  }
  int main ()
  {
  A obj1 // making an object of the class A
  // now we are going to call functions
  obj1.func (2);
  obj1.func (2.65);
  obj1.func (2.5);
  return 0;
  }
```

Now when you run this function, the result will be:

1. 2

2. 2.65
3. 7

As you can see, the **func ()** function is used in 3 different ways.

Operator overload (Operator Overloading):

In C ++, an operator can have more than one meaning.

For example, "+" can be used for both addition and join.

Matching basically means taking two strings and combining them into one.

So 3 + 4 = 7

AND

Block + geeks = Blockgeeks.

The same operator but performs two different functions. This is called Operator Overload.

Polymorphism at compile time is very helpful in blockchain development. It places separate responsibilities for each component and performs many different functions, then in turn, promotes the performance of the entire system.

Feature 5: Isolate the code

C ++ has namespace features, which can be imported from one program to another. Namespace helps avoid name duplication. Also, because C ++ has classes, it can act as boundaries between different APIs and help in explicitly separating.

A C ++ class is a user-defined type or data structure declared with the class keyword, containing functions and data. You can access functions declared in the class by declaring objects of that class.

Feature 6: Calculating development

The programming language can both be self-developed and updated regularly. There are at least 3 compilers and lots of new features to solve practical problems. Debugger - Debugger - and analysis tools are available for everything, from performance profiles to automatic detection of problems. This means that the language is constantly evolving to incorporate newer and better features.

Because of the above features, Satoshi Nakamoto chooses C ++ as the base language of bitcoin source code.

See more:

1. Programming blockchain part 2: Javascript programming language