

solidity自学：第四天

目录

- 内存与区块链: storage与memory
 - 结构体定义与初始化
 - 结构体作为函数参数

内存与区块链: storage与memory

在函数体内定义一个可变长度数组时，它默认的类型是storage，地址拷贝

```
1 function test(uint[] array) returns(uint){
2     arrx = array;//将内存的arry拷贝到区块链上的arrx变量
3     //当我们在函数体内部定义了一个可变长度数组时，实际上，它默认的类型是storage类型，
4     //它指向了区块链上的arrx。
5     //所以当我们修改Z的元素的时候，我们实际上在操作区块链上的arrx
6     uint[] storage Z = arrx;//通过指针实际上修改了区块链上arrx的值
7     Z[0] = 100;
8     //通过指针实际上修改了区块链上arrx的长度，说明Z和arrx其实是一样的。
9     //操作Z的时候，会改变arrx的值
10    Z.length = 1000;
11 }
```

结构体 定义与初始化

```
1 struct student{
2     uint grade;
3     string name;
4
5     mapping(uint => string) map;//在初始化结构体的时候可以忽略掉
6 }
7 student aa;
8 function Init() returns(uint,string,string){
9     //student s = student(100,"Mike");
10    //这里【student(100,"Mike")】默认的是memory类型，它无法赋值给storage类型的student s
11    //应该改成:
12    student memory s = student(100,"Mike");
13    //s.map[52] = "helloWorld"; 无法这么做
14    //原因: memory的对象不可以直接操作struct结构体中的mapping
15    //解决方法如下:
16    aa = s;
17    aa.map[10] = "helloWorld";
18    return(s.grade,s.name,aa.map[10]);
19    //结果: "0": "uint256: 100",
20    //      "1": "string: Mike",
21    //      "2": "string: helloWorld"
22 }
```

结构体作为函数参数

```
1 struct student{
2     uint grade;
3     string name;
4 }
5 //如果形参是struct类型，那么一定要加internal修饰符
6 function test(student s) internal{}
```

storage转storage

```
1 struct student{
```



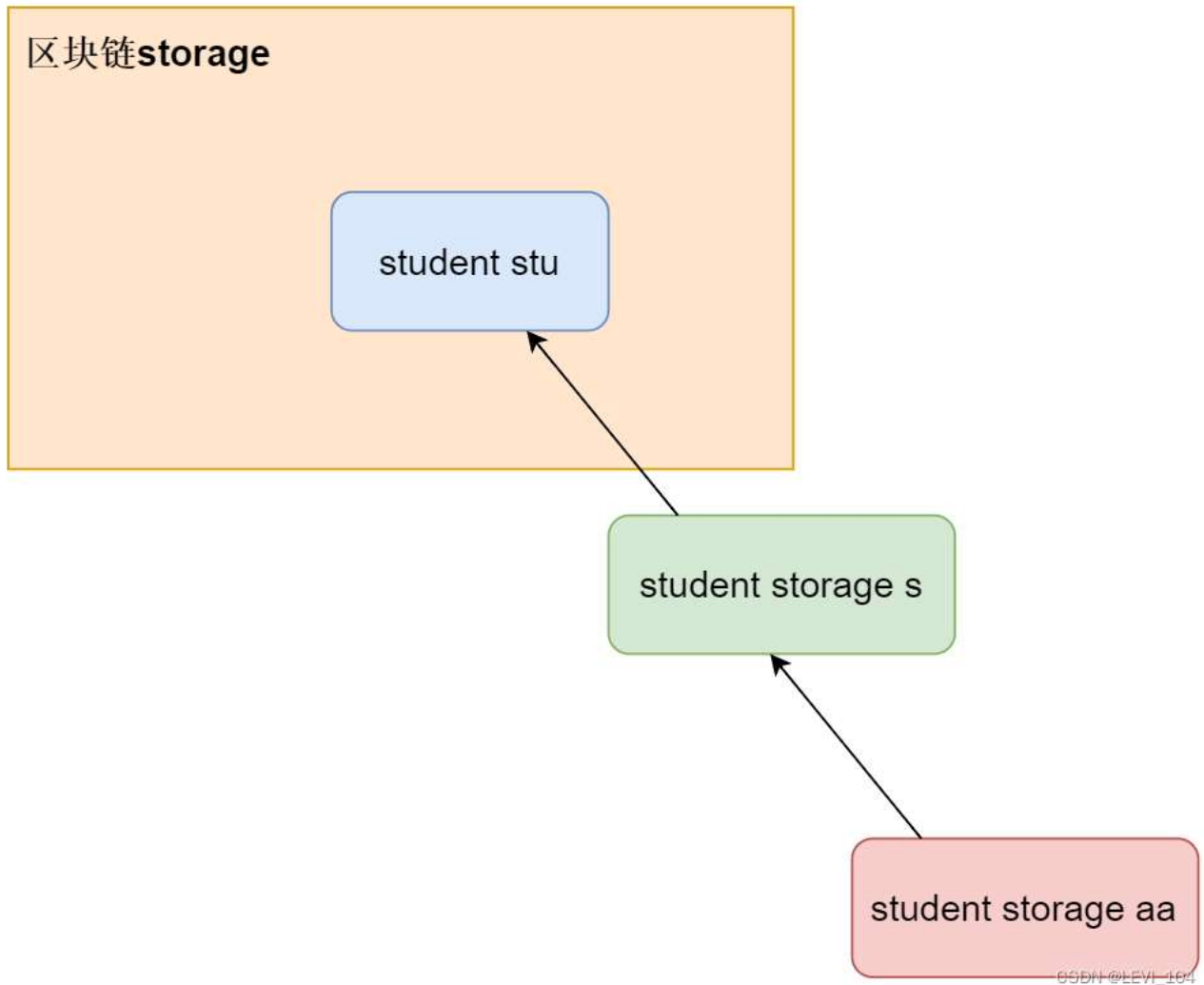
LEVI_104

👍 0 🗨 0 ⭐ 0

```

2      uint grade; 3      string name;
4  }
5  student stu; //这里是在区块链中开辟了一个storage
6  //如果形参是struct类型，那么一定要加internal修饰符
7  function test(student storage s) view internal{
8      //student storage s: s是指针，用来指向区块链storage中的stu
9      //      然后下面修改就是修改指针指向的stu
10     student storage aa = s;
11     aa.name = "陈钦";
12 }
13 function call() returns(string){
14     test(stu);
15     return stu.name;
16 }

```



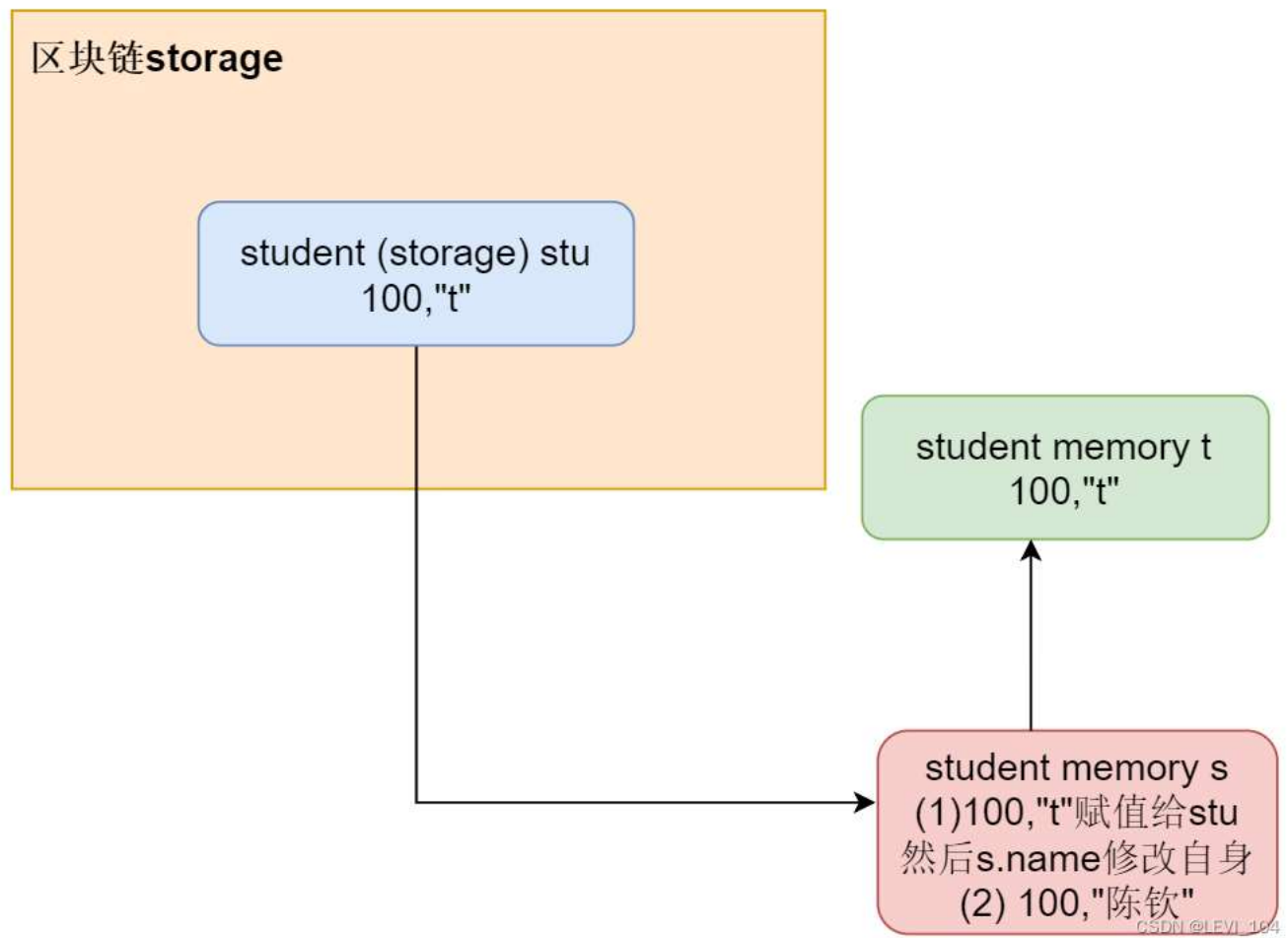
memory转storage

```

1  struct student{
2      uint grade;
3      string name;
4  }
5  student stu; //这里是在区块链中开辟了一个storage
6  //如果形参是struct类型，那么一定要加internal修饰符
7  function test(student memory s) view internal returns(string){
8      stu = s;
9      s.name = "陈钦";
10 }
11 function call() returns(string){
12     student memory t = student(100, "t");
13     test(t);
14     return stu.name;

```

15 | }



storage转memory

```

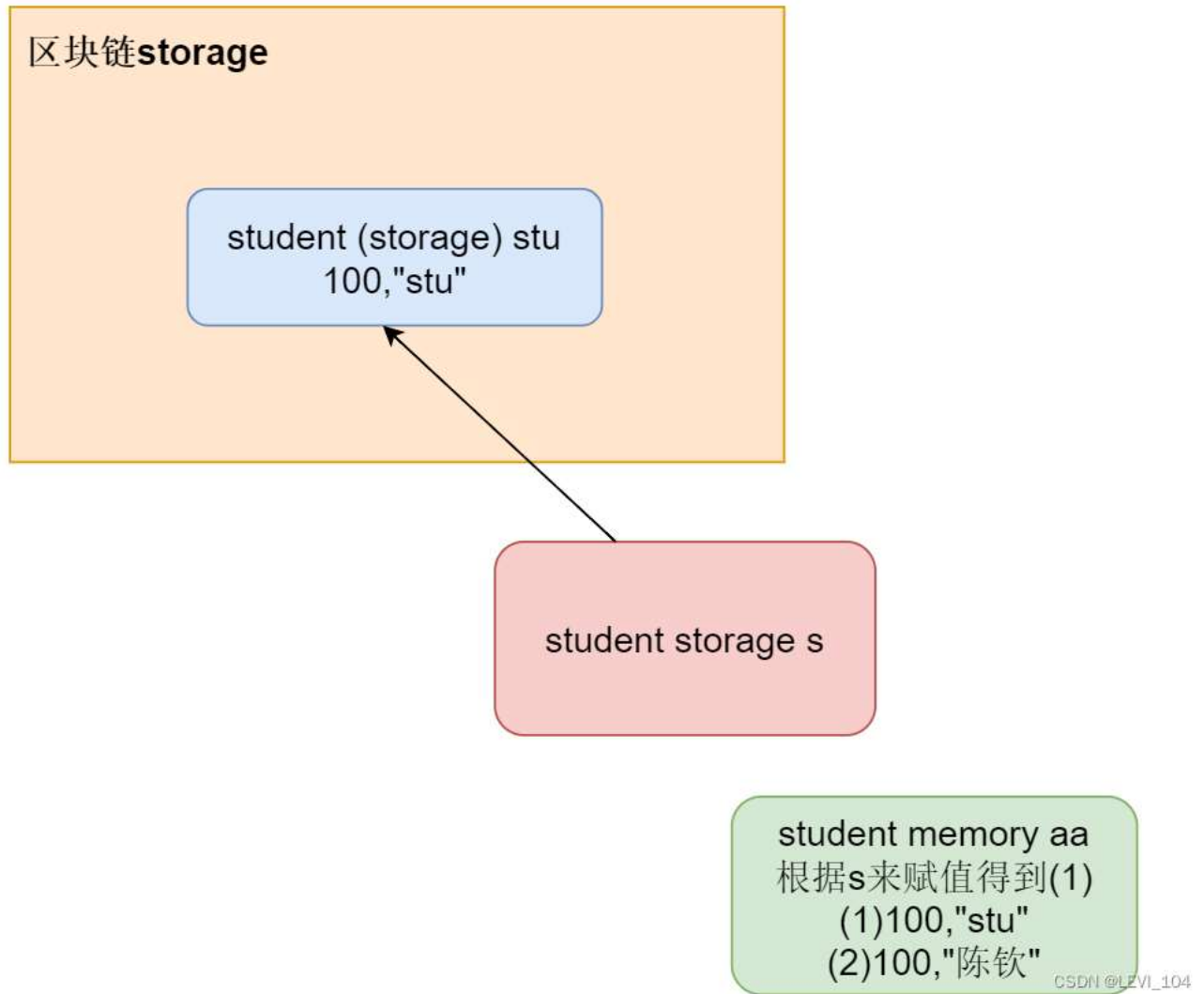
1  struct student{
2      uint grade;
3      string name;
4  }
5  student stu = student(100, "stu");//这里是在区块链中开辟了一个storage
6  //如果形参是struct类型，那么一定要加internal修饰符
7  function test(student storage s) view internal returns(string){
8      //student storage s: s是指针，用来指向区块链storage中的stu
9      //          然后下面修改就是修改指针指向的stu
10     student memory aa = s;
11     aa.name = "陈钦";
12 }
13 function call() returns(string){
14     test(stu);
15     return stu.name;
16 }

```



LEVI_104

👍 0 🗳 0 💬 0 ⭐ 0



memory转memory

```

1  struct student{
2      uint grade;
3      string name;
4  }
5  //如果形参是struct类型，那么一定要加internal修饰符
6  function test(student memory s) view internal returns(string){
7      student memory Mike = s;
8      Mike.name = "陈钦";
9  }
10 function call() returns(string){
11     student memory John = student(100, "John");
12     test(John);
13     return John.name;
14 }

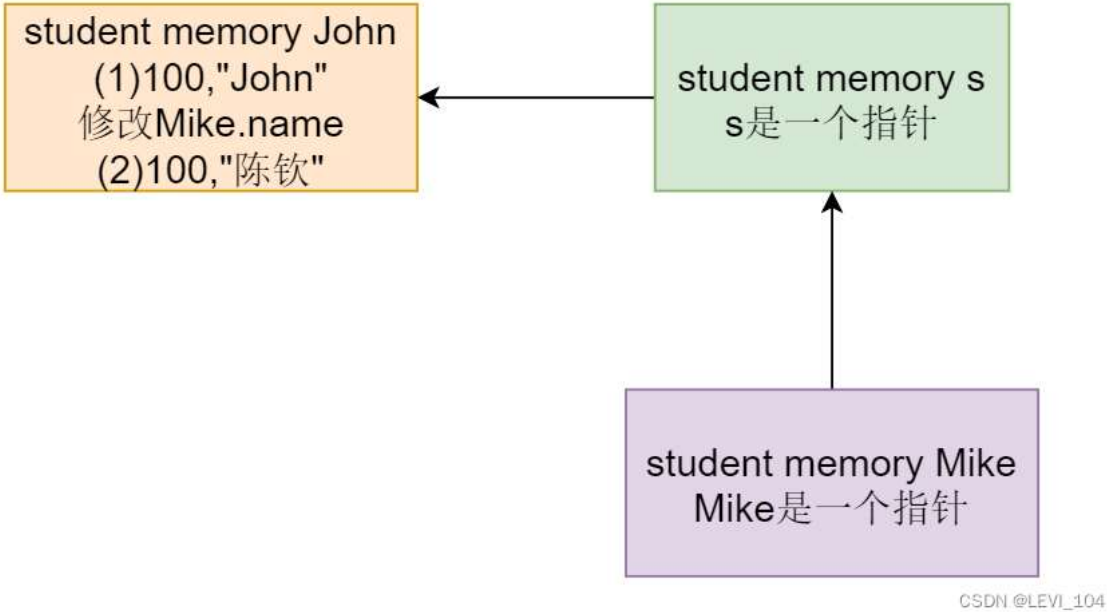
```



LEVI_104

👍 0 🗨 0 ⭐ 0

这是solidity中的优化：struct作为函数体
传来传去非常的不方便，所以在memory的传递
中，用指针来传递



“相关推荐”对你有帮助么？

😞 非常没帮助 😐 没帮助 😐 一般 😊 有帮助 😄 非常有帮助

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00
公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心
家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 ©1999-2022北京创新乐知网络技术有限公司 版权与免责声明 版权申诉
出版物许可证 营业执照