以下是使用Go语言编写的一个简单的转账智能合约，代码中有详细的注释：

```
package mainimport ( "fmt" "github.com/hyperledger/fabric-contract-api-go/contractapi")// ?????????type TransferContract struct { contractapi.Contract}// ??????????????????????????func (tc *TransferContract) Transfer(ctx contractapi.TransactionContextInterface, from, to string, amount int) error { // ?????????? if amount <= 0 { return fmt.Errorf("???? %d ???", amount) } // ????????? fromAccount, err := ctx.GetStub().GetState(from) if err != nil { return fmt.Errorf("???? %s ??", from) } // ????????????? if fromAccount == nil { return fmt.Errorf("?? %s ???", from) } // ????????????? fromBalance := int(fromAccount[0]) // ???????????? if fromBalance < amount { return fmt.Errorf("?? %s ????", from) } // ????? fromBalance -= amount if fromBalance == 0 { // ?????0?????????? err = ctx.GetStub().DelState(from) if err != nil { return fmt.Errorf("???? %s ??", from) } } else { // ??????0??????????? err = ctx.GetStub().PutState(from, []byte{byte(fromBalance)}) if err != nil { return fmt.Errorf("???? %s ??", from) } } // ???????? toAccount, err := ctx.GetStub().GetState(to) if err != nil { return fmt.Errorf("???? %s ??", to) } // ???????????????? if toAccount == nil { err = ctx.GetStub().PutState(to, []byte{byte(amount)}) if err != nil { return fmt.Errorf("???? %s ??", to) } } else { // ????????????????????? toBalance := int(toAccount[0]) toBalance += amount err = ctx.GetStub().PutState(to, []byte{byte(toBalance)}) if err != nil { return fmt.Errorf("???? %s ??", to) } } // ?????? fmt.Printf("?? %s ?? %d ?? %s ??\n", from, amount, to) return nil}func main() { // ??????? transferContract := new(TransferContract)// ??????cc, err := contractapi.NewChaincode(transferContract)if err != nil { return fmt.Errorf("??????: %v", err)}// ???????if err := cc.Start(); err != nil { return fmt.Errorf("?????????: %v", err)}return nil
```

以上代码中，我们创建了一个名为`TransferContract`的智能合约，包含一个`Transfer`方法，用于实现转账功能。在该方法中，我们首先检查转账金额是否合法，然后读取转出账户的余额，更新转出账户和转入账户的余额，并输出转账信息。最后，我们使用`contractapi.NewChaincode`创建了一个新的链码，并使用`cc.Start()`

方法启动了链码服务器。

请注意，此示例仅用于演示目的。在实际开发中，您需要根据您的实际需求编写更复杂的智能合约，并遵循Fabric的安全最佳实践。