

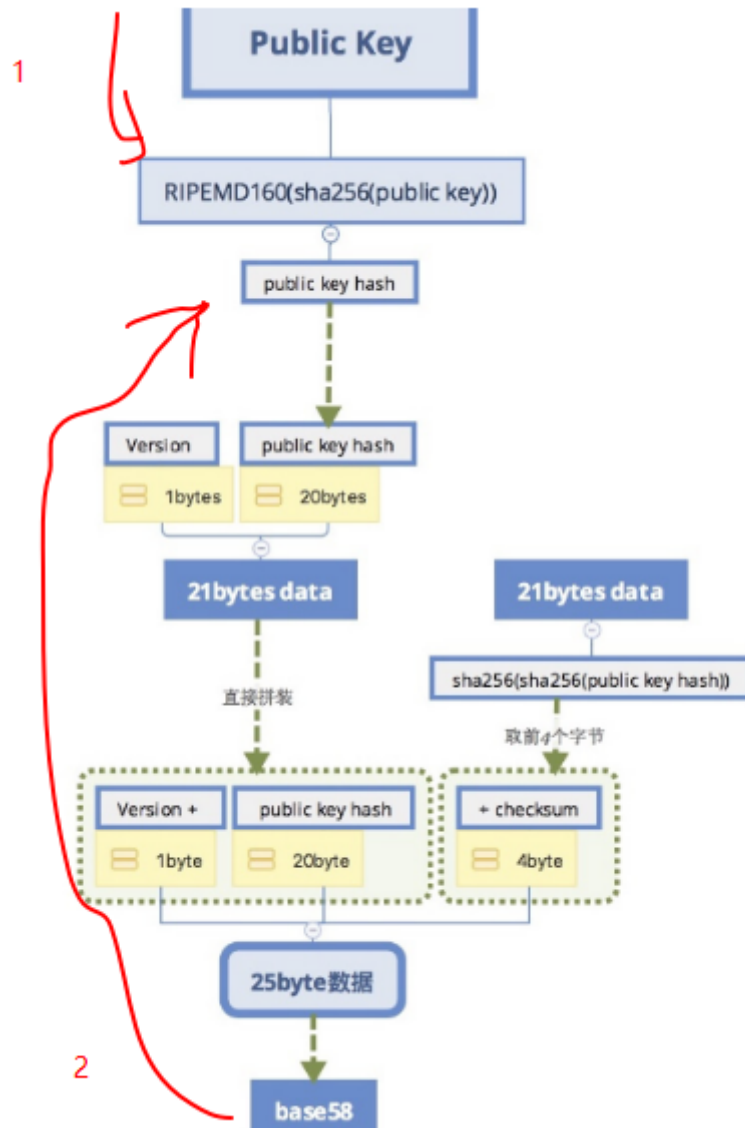
地址改写程序

1. 通过公钥， 获取公钥哈希

1. getPubKeyHashFromPubKey(公钥)

2. 通过地址， 获取公钥哈希

1. getPubKeyHashFromAddress (地址)



实现三个函数：

```
//1. 通过公钥， 获取公钥哈希
func getPubKeyHashFromPubKey(pubKey []byte) []byte {
    //一、第一次哈希
    firstHash := sha256.Sum256(pubKey)
    //第二次哈希
    hasher := ripemd160.New()
```

```

hasher.write(firstHash[:])
pubKeyHash := hasher.Sum(nil)

return pubKeyHash
}

//2. 通过地址，获取公钥哈希
func getPubKeyHashFromAddress(address string) []byte {
    //1. base58解码，得到25字节数据
    decodeInfo := base58.Decode(address)

    if len(decodeInfo) != 25 {
        fmt.Println("地址长度无效，应该为25字节，当前字节为:", len(decodeInfo))
        return nil
    }

    //2. 截取出中间的20字节
    pubKeyHash := decodeInfo[1:21]

    //3. 返回
    return pubKeyHash
}

func checksum(payload []byte) []byte {
    f1 := sha256.Sum256(payload)
    second := sha256.Sum256(f1[:])

    //checksum := second[:] //作闭右开
    checksum := second[:4] //作闭右开
    return checksum
}

```

改写地址函数

```

func (w *wallet) getAddress() string {
    //通过公钥， 获取公钥哈希
    pubKeyHash := getPubKeyHashFromPubKey(w.PubKey)

    //二、在前面添加1个字节的版本号
    payload := append([]byte{byte(00)}, pubKeyHash...)

    //三、做两次哈希运算，截取前四个字节，作为checksum，
    checksum := checksum(payload)

    //四、拼接25字节数据
    payload = append(payload, checksum...)

    //五、base58处理，得到地址
    address := base58.Encode(payload)
    return address
}

```

技巧：Ctrl+R，输入“cre”，此时会显出最近的含义cre开头的命令，
如果不是想要的，可以继续ctrl +R，此时会查找另外一个cre开头的命令。

```
duke@DUKEDU51C6 MINGW64 /c/goprojects/src/go5期/03-比特币/v5-a-wallet  
(reverse-i-search)`':
```

终端常用操作：

1. ctrl + a => 移动到行首
2. ctrl + e => 移动到行尾

1. ctrl + f =》 向前移动一个字符
2. ctrl + b=》 向后移动一个字符

1. ctrl + h=》 向前删除一个字符
2. ctrl + d=》 向后删除一个字符

1. ctrl + u =》 删除当前光标前的所有字符
2. ctrl + k=》 删除当前光标后的所有字符
3. ctrl + l =》 清屏

改写交易

```
//交易输入  
type TXInput struct {  
    //1. 所引用的output所在的交易id  
    TXID []byte  
    //2. 所引用的output的索引值  
    Index int64  
    //3. 解锁脚本：  
    //ScriptSig string //先使用string代替，后续会改成签名  
  
    //1. 私钥签名  
    ScriptSig []byte    <<====  
    //2. 公钥  
    PubKey []byte    <<====  
}  
  
//交易输出  
type TXOutput struct {  
    //1. 锁定脚本  
    //LockScript string
```

```

//1收款人的公钥哈希
PubKeyHash []byte <<=====
//2. 转账金额
value float64
}

```

提供NewTXOutput方法

```

//收款人给付款人地址，锁定的时候不是使用地址锁定的，而是使用公钥哈希锁定的
//提供一个生成output的方法
func NewTXOutput(value float64, address string) TXOutput {
    //计算公钥哈希
    pubKeyHash := getPubKeyHashFromAddress(address)

    output := TXOutput{
        PubKeyHash: pubKeyHash,
        value:      value,
    }

    return output
}

```

改写挖矿交易：

```

func NewCoinbaseTx(miner string, data string) *Transaction {
    inputs := []TXInput{{
        TXID:      nil,
        Index:     -1,
        ScriptSig: []byte(data), <<====
        PubKey:    nil, <<====
    }}

    output := NewTXOutput(reward, miner) <<====
    outputs := []TXOutput{output} <<====

    //outputs := []TXOutput{{
    //  LockScript: miner,
    //  value:      reward,
    //}}

    tx := &Transaction{
        TxInputs:  inputs,
        TXOutputs: outputs,
        Timestamp: time.Now().Unix(),
    }

    //设置交易id
    tx.SetTxId()

    return tx
}

```

改写NewTransaction

```
func NewTransaction(from, to string, amount float64, bc *BlockChain) (*Transaction, error) {  
  
    //1. 打开钱包  
    wm := NewWalletManager()  
    if wm == nil {  
        return nil, errors.New(text: "打开钱包失败!")  
    }  
  
    //2. 找到付款方对应的私钥和公钥  
    w, ok := wm.Wallets[from]  
    if !ok {  
        return nil, fmt.Errorf(format: "没有找到: '%s'对应的钱包!", from)  
    }  
    //创建input的时候需要私钥签名和公钥  
    //priKey := w.PrivKey //TODO 汇通在使用  
    pubKey := w.PubKey  
  
    //付款人的公钥哈希  
    pubKeyHash := getPubKeyHashFromPubKey(pubKey)  
  
    //1. 1. 找到付款人能够支配的合理的钱, 返回金额和utxoinfo
```

```
    //1. 1. 找到付款人能够支配的合理的钱, 返回金额和utxoinfo  
    utxoinfos, value := bc.FindNeedUtxoInfo(pubKeyHash, amount)  
  
    //2. 判断返回金额是否满足转账条件, 如果不满足, 创建交易失败。  
    if value < amount {  
        return nil, errors.New(text: "付款人金额不足!")  
    }  
}
```

```
    //1. 拼装inputs  
    for _, utxoinfo := range utxoinfos {  
        input := TXInput{  
            TXID:      utxoinfo.txid,  
            Index:     utxoinfo.index,  
            ScriptSig: nil, //钱包再交易创建的最后再处理 TODO  
            PubKey:    pubKey, //付款人的公钥  
        }  
  
        inputs = append(inputs, input)  
    }  
}
```

```
//1. 拼装一个属于收款人的output
output := NewTXOutput(amount, to)

//output := TXOutput{
//  LockScript: to,
//  Value:      amount,
//}
outputs = append(outputs, output)

//2. 判断一下是否需要找零, 如果有, 拼装一个属于付款方output
if value > amount {
    //找零
    //output1 := TXOutput{
    //  LockScript: from,
    //  Value:      value - amount,
    //}
    output1 := NewTXOutput(value-amount, from)
    outputs = append(outputs, output1)
}
```

- TXInput
- TXOutput
- newTXOutput方法
- NewCoinbaseTx
- NewTransaction
- FindNeedUTXO改写
- FindMyUtxo改写
- getBalance改写

校验地址有效性

签名相关

校验签名
