**深 圳 大 学 实 验 报 告**

**课程名称：­ 编程语言**

**实验项目名称： 编程语言特性及其应用**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 许智武**

**报告人： 林佳涛 学号：2016150095 班级： 01**

**实验时间： 2018年3月15日至2018年4月30日**

**实验报告提交时间： 2018年4月28日**

**教务部制**

|  |  |  |
| --- | --- | --- |
| **实验目的与要求：**  **目的：**采用语言的特性来解决计算机工程问题，包括结构的设计与应用、递归和迭代的设计与应用、mapreduce的设计与应用等实验。通过语言特性的应用实验，熟练掌握编程语言的特性及其编程思想，如数据抽象、递归和迭代的设计、分而治之等思想，培养使用不同语言特性解决复杂计算机工程问题，并对结果进行分析的能力  **要求:**  本实验分为三部分，如下所示  **第一部分：结构的设计与应用**  请用你熟悉的语言实现一个简易区块链系统, 功能包括查询块长度，插入块，查找块，检查块信息等，请分别用记录结构（如C语言的struct、Java的class等）和不用记录结构实现，并比较（如可读性、可写性、可靠性、效率等）。  提示：  块结构的信息：索引，时间戳，数据，哈希值，前置哈希值  块的生成：注意首块和其他块的区别  块的存储结构（账本）：块链表  检查块：前后索引一致？前后hash值一致？当前hash值正确？  **第二部分：递归和迭代的设计与应用**  大部分语言都提供了递归与迭代这两种机制，例如，最大公约数的计算可分别用这两种机制来实现：  /\* 递归实现 \*/  int gcd(int a, int b) {  /\* assume a, b > 0 \*/  if (a == b) return a;  else if (a > b) return gcd(a-b, b);  else return gcd(a, b-a);  }  /\* 迭代实现 \*/  int gcd2(int a, int b) {  /\* assume a, b > 0 \*/  while (a != b) {  if (a > b) a = a-b;  else b = b-a;  }  return a;  }  一、请用你熟悉的语言分别用递归和迭代这两种机制来设计和解决以下两个问题之一，并对结果进行分析：   1. N皇后问题：在n×n格的国际象棋上摆放n个皇后，使其不能互相攻击的摆法。 2. 全排列生成问题：输入一个字符串，输出改字符串中字符的所有排列。   二、请分析满足什么条件的递归能转化为迭代以及如何转化。（选做）实现递归转迭代的转化器。  **第三部分：mapreduce的设计与应用**  Google处理大规模数据的mapreduce来源于函数语言的map函数和reduce函数，下面是map和reduce的js实现：  /\* map的js实现 \*/  function map (f, inarray) {  var out = [];  for(var i = 0; i < inarray.length; i++) {  out.push( f(inarray[i]) )  }  return out;  }  /\* reduce的js实现 \*/  function reduce (f, inarray) {  if(inarray.length <= 1) return;  if(inarray.length == 2) return f(inarray[0],inarray[1]);  r = inarray[0];  for(var li = 1;li < inarray.length; li++) {  r = f(r, inarray[li]);  }  return r;  }  简单来说，map将函数f分别作用到数组inarray的每个元素上，并返回由这些作用结果组成的新数组，而reduce将函数f分别从左到右地作用到数组inarray的（两个）元素，并返回最后的作用结果，如下列例子所示：  js> map(function(x){return x+1}, [1,2,3,4,5])  [2,3,4,5,6]  js> reduce( function(x,y){return x+y}, [1,2,3,4,5])  15  js> reduce( function(x,y){return x\*y}, [1,2,3,4,5])  一、请实现map和reduce函数（建议使用支持函数作为参数的语言，如Haskell、js、python等），并使用map和reduce来   1. 计算数组的平方和，如[1,2,3,4]的平方和为12+22+32+42。 2. 统计数组中正数的个数，如[-1,1,0,-2,5]的正数个数为2。 3. 展平数组的数组，如[[1,2],[3,4,5],[6,[7]]]展平后为[1,2,3,4,5,6,7]。   二、现实世界中数据可能有各种不同的结构，二叉树是其中常用的一种结构。请设计能够处理二叉树数据的maptree和reducetree。   1. 请实现maptree和reducetree 2. 请使用maptree和reducetree为下列通讯录加上区号，并统计深圳电话的个数     提示：假设有树结构的抽象数据类型：  data Tree type = Leaf type | Node (Tree type) (Tree type)  如maptree f(x){return x+1} (Node (Node (Leaf 1) (Leaf 2)) (Leaf 3))  = (Node (Node (Leaf 2) (Leaf 3)) (Leaf 4))  如reducetree f(x,y){return x+y} (Node (Node (Leaf 1) (Leaf 2)) (Leaf 3))  = (1+2)+3 = 6 | | |
| 方法、步骤：  本实验分为三部分，如下所示  **第一部分：结构的设计与应用**  使用C++实现一个简易区块链系统包含以下功能：   1. 功能包括 2. 查询块长度 3. 插入块 4. 查找块 5. 检查块信息   请分别用记录结构（如C语言的struct、Java的class等）和不用记录结构实现，并比较（如可读性、可写性、可靠性、效率等）。  记录结构：  图片包含 屏幕截图  已生成极高可信度的说明  **未记录结构：**  **图片包含 文字, 屏幕截图  已生成高可信度的说明**  **第二部分：递归和迭代的设计与应用**  分别用递归和迭代这两种机制来设计和解决以下两个问题之一，并对结果进行分析：   1. N皇后问题：在n×n格的国际象棋上摆放n个皇后，使其不能互相攻击的摆法。 2. 全排列生成问题：输入一个字符串，输出改字符串中字符的所有排列。   二、请分析满足什么条件的递归能转化为迭代以及如何转化。（选做）实现递归转迭代的转化器。  **第三部分：mapreduce的设计与应用**  map将函数f分别作用到数组inarray的每个元素上，并返回由这些作用结果组成的新数组，而reduce将函数f分别从左到右地作用到数组inarray的（两个）元素，并返回最后的作用结果。  实现map和reduce函数（建议使用支持函数作为参数的语言，如Haskell、js、python等），并使用map和reduce来：   1. 计算数组的平方和，如[1,2,3,4]的平方和为12+22+32+42。 2. 统计数组中正数的个数，如[-1,1,0,-2,5]的正数个数为2。 3. 展平数组的数组，如[[1,2],[3,4,5],[6,[7]]]展平后为[1,2,3,4,5,6,7]。   二、现实世界中数据可能有各种不同的结构，二叉树是其中常用的一种结构。请设计能够处理二叉树数据的maptree和reducetree。   1. 实现maptree和reducetree 2. 使用maptree和reducetree为下列通讯录加上区号，并统计深圳电话的个数 | | |
| 实验过程及内容：  **第一部分：结构的设计与应用**  使用C++实现一个简易区块链系统包含以下功能：   1. 功能包括 2. 查询块长度 3. 插入块 4. 查找块 5. 检查块信息   首先实现一个区块的结构：  如图区块的结构中包含前一个区块的哈希值Prehash，数据Data，时间戳TimeStamp以及区块索引Index；    由于区块具有前置哈希的属性，而区块链是一条链，如果一条链不是循环链的话，他总有一个区块头，so问题来了，区块链的头的前置哈希值需要如何定义。结论是：人工定义。其中区块的头叫做创世块Genesis Block。而在这其中我没有定义区块的hash值，是由于我想为了方便定义创世块，而将区块的hash值放置在BlockHead的子类中去。以下是Block类：  图片包含 文字  已生成高可信度的说明  请分别用记录结构（如C语言的struct、Java的class等）和不用记录结构实现，并比较（如可读性、可写性、可靠性、效率等）。  ***记录结构：***  我所定义的区块链结构为：  ChainClass+BlockClass；  Chain为一条区块链，其中包含了若干个前后连接的区块Block；Chain对每一个属性有很好的封装，vector为一条Block链，其中还包含了遍历输出链，获取链的长度，添加区块，查找区块，检查区块信息是否准确。    其中我们先来看看区块的定义：  图片包含 就坐  已生成高可信度的说明  上面对区块class的定义仅仅只是头文件。主要的实现工作是在Block.cpp中：  图片包含 文字  已生成极高可信度的说明  Block的定义需要是每创建一个Block值，就需要获取当前的时间戳以及计算当前的hash值。  图片包含 屏幕截图  已生成高可信度的说明  这里定义的是一个根据索引前一个区块的哈希值以及数据还有时间戳来计算当前区块的哈希值，其中的计算方法hash256是我从网上找到的算法，将前置哈希，数据，时间戳以及索引按字符串拼接起来再放到hash256中去计算，得到返回的结果。  图片包含 文字  已生成高可信度的说明  此处是使用内置时间模块<time.h>获取系统当前的时间。并按照年月日时分秒排序  图片包含 文字  已生成高可信度的说明  此处是一个数据的写入。被Chain中的成员函数调用。  Chain中详细的头文件信息如下：  图片包含 屏幕截图  已生成极高可信度的说明  其中各自的实现在：  图片包含 文字  已生成高可信度的说明  如上图所示，初始化一个区块链的同时即创建好了一个创世块。这里直接给区块的内容直接赋值，这是人工操作的行为。  其中添加块的功能如下：  图片包含 文字  已生成高可信度的说明  此处由两个函数共同完成，add\_block是public类型的成员函数。    此处是获取链的长度。  图片包含 屏幕截图  已生成高可信度的说明此处是使用一个简单的循环语句进行遍历查找。按索引匹配。  图片包含 屏幕截图  已生成高可信度的说明  此处是检查区块的信息是否正确，这一部分需要完成的是查找每一个区块的前置hash是否与前一个区块的hash值相符合。如果不符合即可判断当前区块非法。  图片包含 屏幕, 黑色  已生成高可信度的说明  此处是输出所有的区块信息，找出了每一个区块的信息并打印出来  其中完整的mian函数如下：  图片包含 屏幕截图, 文字  已生成高可信度的说明  运行结果为：  插入：    查找：  图片包含 屏幕截图  已生成高可信度的说明  图片包含 屏幕截图  已生成极高可信度的说明  检查：    查询长度：  图片包含 屏幕截图  已生成高可信度的说明  输出所有值：  图片包含 屏幕截图  已生成极高可信度的说明  ***未记录结构：***  完整的属性以及成员函数如下：    该部分只包含了non-structure-blockchain.h和non-structure-blockchain.cpp两个文件。  其中主要的结构为一个hash值数组数据数组时间戳数组。  图片包含 文字  已生成高可信度的说明  主函数的实现如上，其中同样的需要创建创世块，max作为索引index。代表了下一个块插入的索引值，也代表了链的长度。    非结构化的区块链天加快比较简单，仅仅只是计算出时间戳以及区块的hash值再赋值即可。    计算哈希值，根据索引前一个块的hash值以及当前块创建的时间戳来计算得到。  图片包含 文字  已生成高可信度的说明  这里是利用系统内置模块获取时间。    查找块，同样的是顺序索引查找，    长度为块的长度。  图片包含 文字  已生成高可信度的说明  遍历区块链，输出所有数据；  其中主函数与结构化的类似：  图片包含 屏幕截图, 文字  已生成高可信度的说明  实验结果如下：  插入：  图片包含 屏幕截图  已生成高可信度的说明  查找：  图片包含 屏幕截图  已生成极高可信度的说明  检查：    输出：  图片包含 屏幕截图, 文字  已生成高可信度的说明  对比了这两种方法发现：  结构化的可读性，易修改性，安全性都高于非结构化，对每一个属性或者函数都有一个完成的描述和封装。修改起来方便了许多。  **第二部分：递归和迭代的设计与应用**  分别用递归和迭代这两种机制来设计和解决以下两个问题之一，并对结果进行分析：   1. N皇后问题：在n×n格的国际象棋上摆放n个皇后，使其不能互相攻击的摆法。   递归：  主函数：  图片包含 文字  已生成高可信度的说明  递归主体：    判断是否安全：  图片包含 文字  已生成高可信度的说明  输出结果：  图片包含 文字  已生成极高可信度的说明  实验结果： 分别看4 , 6 , 8皇后：  图片包含 电子产品  已生成高可信度的说明  图片包含 电子产品, 计算机, 键盘  已生成极高可信度的说明    迭代：  主函数：  图片包含 设备  已生成高可信度的说明  迭代入口：  图片包含 户外  已生成高可信度的说明  迭代主体：  图片包含 文字  已生成极高可信度的说明  判断是否安全：    打印结果：  图片包含 文字, 时钟  已生成高可信度的说明  实验结果：  分别尝试4 6 8 皇后：  图片包含 文字  已生成极高可信度的说明    图片包含 文字  已生成高可信度的说明   1. 全排列生成问题：输入一个字符串，输出改字符串中字符的所有排列。   递归：  主函数：  图片包含 文字  已生成极高可信度的说明  递归主体：  图片包含 文字  已生成高可信度的说明  实验结果：      迭代：  主函数：  图片包含 设备  已生成高可信度的说明  迭代主体：  图片包含 文字  已生成极高可信度的说明  迭代内容：  图片包含 文字  已生成高可信度的说明  交换函数：  图片包含 文字  已生成高可信度的说明  拼接函数：  图片包含 物体, 时钟  已生成极高可信度的说明  实验结果：  图片包含 设备, 仪表  已生成高可信度的说明  二、请分析满足什么条件的递归能转化为迭代以及如何转化。（选做）实现递归转迭代的转化器。  答：从理论上讲，只要允许使用栈，所有的递归程序都可以转化成迭代。但是并非所有递归都必须用栈，不用堆栈也可以转化成迭代的，大致有两类：  尾递归：可以通过简单的变换，让递归作为最后一条语句，并且仅此一个递归调用。自顶向下->自底向上：对程序的结构有深刻理解后，自底向上计算，比如 fibnacci 数列的递归->迭代转化。**第三部分：mapreduce的设计与应用**  map将函数f分别作用到数组inarray的每个元素上，并返回由这些作用结果组成的新数组，而reduce将函数f分别从左到右地作用到数组inarray的（两个）元素，并返回最后的作用结果。  实现map和reduce函数（建议使用支持函数作为参数的语言，如Haskell、js、python等），并使用map和reduce来：  Map：  图片包含 屏幕截图  已生成高可信度的说明  测试：    结果：    Reduce:  图片包含 屏幕截图  已生成极高可信度的说明  主函数：    测试结果：     1. 计算数组的平方和，如[1,2,3,4]的平方和为12+22+32+42。   图片包含 屏幕截图  已生成极高可信度的说明  测试结果：     1. 统计数组中正数的个数，如[-1,1,0,-2,5]的正数个数为2。   图片包含 屏幕截图  已生成高可信度的说明  图片包含 屏幕截图  已生成高可信度的说明  将其转化为01串，再累加，实验结果如下：     1. 展平数组的数组，如[[1,2],[3,4,5],[6,[7]]]展平后为[1,2,3,4,5,6,7]。   自己定义一个递归函数铺平：    其中调用的为：    测试结果如下：    二、现实世界中数据可能有各种不同的结构，二叉树是其中常用的一种结构。请设计能够处理二叉树数据的maptree和reducetree。   1. 实现maptree和reducetree   MapTree  图片包含 文字  已生成极高可信度的说明  利用异常处理进行的递归调用。测试结果：    图片包含 屏幕截图  已生成极高可信度的说明  ReduceTree  图片包含 文字  已生成极高可信度的说明  主函数：    结果如下：     1. 使用maptree和reducetree为下列通讯录加上区号，并统计深圳电话的个数     定义好树：    增加处理节点函数：  加区号  找出深圳的标记为1  总的函数为图片包含 文字  已生成高可信度的说明  Main函数  输出结果：  图片包含 屏幕截图  已生成极高可信度的说明 |

深圳大学学生实验报告用纸

|  |
| --- |
| 实验结论：  通过本次实验，查阅了许多的相关资料，从第一题来说，区块链对我来说是一个熟悉但陌生的存在，在近几年区块链行业的突飞猛进，各种大会和新闻都传入耳里，但我们对此的了解仅仅停留在概念上，而不知区块链真正的面貌是如何。从本次实验学习到，区块链本质上也如同链表一样。是一条链，但不同的是他的安全性和可靠性是十分强大的，我们也通过实验了解到了这一点。其次在迭代和递归的实验中，迭代和递归的相互转换的要点让人捉摸了许久，最后虽然勉强自己得出了一个结论，循环和递归的相互转化等浅显的结论，通过阅读文献资料，学习到了其中真正的转化面貌。而在最后的内置函数reduce与map的实现中，这个虽然不是本章的难点，但是如果要做到像python内置模块那样的效果，我们写的这些远远比不上，但是也从这次实验中发现，可以从查看内置模块代码来学习。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。