重庆大学本科学生实验项目任务书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验题目 | 二叉树的应用 | | | |
| 实验时间 | 2021.11.25 | 实验地点 | | DS1404 |
| 实验性质 | □验证性 √设计性 □综合性 | | | |
| 实验目的  1. 学习并掌握二叉树数据结构及实现方法，掌握树的遍历方法。  2. 熟练掌握二叉检索树（BST）的基本操作及其应用。 | | | | |
| 实验内容：   1. 利用BST实现一个城市数据库：每个数据库结点包括城市名称和以整数x与y表示的城市坐标，根据城市名称组织该BST； 2. 在该数据库上实现按城市名称进行的插入、删除和检索； 3. 打印出以指定字母打头的所有城市记录； 4. 打印出与指定点的距离在给定值之内的所有城市记录； 5. 最后提交完整的实验报告和源程序。 | | | | |
| 参考资料：   * Data Structures and Algorithm Analysis (C++ Version) Clifford A. Shaffer   + Data Structure and Algorithm Analysis in C++ (Third Edition)，Mark Allen Weiss， Pearson Education, 2006.   + Data Structures, Algorithms, and Applications in C++，Sartaj Sahni， McGraw-Hill, 1998.   + 《数据结构（ C 语言版）》，严蔚敏，吴伟民编著，清华大学出版社，2007年第1版 | | | | |
| 任务下达日期 2021 年 11月 22 日 | | | 完成日期 2021 年 11月 25 日 | |

说明：学院、专业、年级均填全称，如：计算机学院、计算机科学与技术、2017。

实验报告模板见下页：

**《（课程名称）》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2020级计算机科学与技术卓越一班** | | | **姓名** | **孙莹莹** |
| **实验题目** | 二叉树的应用 | | | | | |
| **实验时间** | **2021.11.25** | | **实验地点** | **DS4104** | | |
| **实验成绩** |  | | **实验性质** | **□验证性 □设计性 □综合性** | | |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 一、实验目的  1. 学习并掌握二叉树数据结构及实现方法，掌握树的遍历方法。  2. 熟练掌握二叉检索树（BST）的基本操作及其应用。 | | | | | | |
| 二、实验项目内容   1. 利用BST实现一个城市数据库：每个数据库结点包括城市名称和以整数x与y表示的城市坐标，根据城市名称组织该BST； 2. 在该数据库上实现按城市名称进行的插入、删除和检索； 3. 打印出以指定字母打头的所有城市记录； 4. 打印出与指定点的距离在给定值之内的所有城市记录； | | | | | | |
| 1. 实验过程或算法（源程序） 2. 设计思路 3. 首先实现二叉树的结点类BSTNode   有成员函数结点的键（key)（用于检索）、值(it)，左右子结点(lc,rc)。成员函数则包括查找及修改键、值(key(),element(),setKey(),setelement())，查找及修改左右子结点(left(),right(),setLeft(),setRight())，判断是否为叶节点(isLeaf())。  template <typename Key, typename E>  class BSTNode  {  private:      Key k;  //node's key      E it;   //node's value;      BSTNode \* lc;  //node's leftchild      BSTNode \* rc;   //node's rightchild  public:      BSTNode() { lc = rc = nullptr; }      BSTNode(Key K, E e, BSTNode \*l = nullptr, BSTNode \*r = nullptr)      {          k = K;          it = e;          lc = l;          rc = r;      }      ~BSTNode() = default; // Destructor      E & element() {return it;}  //get value      void setElement(const E & e) {it = e;}  //set value      Key & key() {return k;}  //get key      void setKey(const Key & K) {k = K;}  //set key      inline BSTNode \* left() const {return lc;}  //find leftchild      void setLeft(BSTNode<Key,E> \*b) {lc = (BSTNode \*)b;}  //set leftchild      inline BSTNode \*right() const {return rc;}  //find rightchild      void setRight(BSTNode<Key,E>\*b){rc = (BSTNode\*)b;} //set rightchild     bool isLeaf() {return (lc==NULL) && (rc==NULL);}  //whether is leaf  };   1. 实现BST树类   有成员根节点（root)及结点个数(nodecount)，成员函数则包括执行插入(insert)，删除(remove)，查找（find),中序遍历输出(print),树的结点个数（size),清空(clear)操作的函数。  template <typename Key, typename E>  class BST  {  private:      BSTNode<Key,E> \* root;   //the root of BST      int nodecount;  //number of nodes in BST      void clearhelp(BSTNode<Key,E> \* root) //delete root and it's descendents      {          if(root==NULL)  return;          clearhelp(root->left());          clearhelp(root->right());          delete root;      }        BSTNode<Key,E> \*inserthelp(BSTNode<Key,E>\*root, const Key &k, const E & it)      {          if(root==NULL)              return new BSTNode<Key,E>(k,it,NULL,NULL);          if(root->key()>k)              root->setLeft(inserthelp(root->left(),k,it));          else              root->setRight(inserthelp(root->right(),k,it));          return root;      }        BSTNode<Key,E> \* getmin(BSTNode<Key,E> \*rt)      {          if(rt->left()==NULL)              return rt;          else              return getmin(rt->left());      }        BSTNode<Key,E> \*deletemin(BSTNode<Key,E> \*rt)      {          if(rt->left()==NULL)              return rt->right();          else          {              rt->setLeft(deletemin(rt->left()));              return rt;          }      }      BSTNode<Key,E> \* removehelp(BSTNode<Key,E> \*rt, const Key &k)      {          if(rt==NULL)              return NULL;          else if(k<rt->key())              rt->setLeft(removehelp(rt->left(),k));          else if(k>rt->key())              rt->setRight(removehelp(rt->right(),k));          else          {              BSTNode<Key,E> \* temp = rt;              if(rt->left()==NULL)              {                  rt = rt->right();                  delete temp;              }              else if(rt->right()==NULL)              {                  rt = rt->left();                  delete temp;              }              else              {                   BSTNode<Key,E> \* temp = getmin(rt->right());                  rt->setElement(temp->element());                  rt->setKey(temp->key());                  rt->setRight(deletemin(rt->right()));                  delete temp;              }            }          return rt;      }        E findhelp(BSTNode<Key,E> \*root, const Key & k)      {          if(root==NULL)              return (E)NULL;          if(k<root->key())              return findhelp(root->left(),k);          else if(k>root->key())              return findhelp(root->right(),k);          else              return root->element();      }          void printhelp(BSTNode<Key, E> \*root, int level) const  //中序输出      {          if (root == nullptr)              return;                         // Empty tree          printhelp(root->left(), level + 1); // Do left subtree          //for (int i = 0; i < level; i++)     // Indent to level          //    cout << "  ";          cout << root->key() << endl;         // Print node value          printhelp(root->right(), level + 1); // Do right subtree      };  public:      BST()      {          root = NULL;          nodecount = 0;      }      ~BST() { clearhelp(root); }        void clear()      {          clearhelp(root);          root = NULL;          nodecount = 0;      }       void insert(const Key &k, const E & e)      {          root = inserthelp(root,k,e);          nodecount++;      }        E remove(const Key & k)      {          E temp = findhelp(root,k);          if(temp!=NULL)          {              root = removehelp(root,k);              nodecount--;          }          return temp;      }        E find(const Key &k) const { return findhelp(root, k); }      BSTNode<Key,E> \* getroot()      {          return root;      }        int size() { return nodecount; }      void print() const      {          if(root==NULL)              cout << "The BST is empty" << endl;          else              printhelp(root, 0);      }      };   1. 实现坐标类（Location)   包括成员横坐标（x),纵坐标(y),该坐标是否有效(isNull),成员函数则重载了！=符号（因为在BST的实现中需要用到！=，因此需要重载）,返回x的值(getx()),返回y的值(gety())，及重载输出<<。  class Location  {  public:      int x;      int y;      bool isNull;        Location(nullptr\_t = nullptr)      {          x = 0x3f3f3f3f;          y = 0x3f3f3f3f;          isNull = true;      }      explicit Location(int xlo, int ylo)      {          x = xlo;          y = ylo;          isNull = false;      }        bool operator!=(Location \*)      {          return !isNull;      }      int getx()      {          return x;      }      int gety()      {          return y;      }      /\*\* 重载输出\*/      friend ostream &operator<<(ostream &out, const Location &l);  };  ostream &operator<<(ostream &out, const Location &l)  {      out << l.x << " " << l.y;      return out;  }   1. 建立城市数据库   用BST<string,Location>去建立以城市名为key,城市坐标为it的二叉树。成员函数包括插入(insert),按首字母检索(printCaptital()),按距离检索(printDistance()),及命令执行函数(runCmd（））。  其中首字母检索及距离检索思路大致相同，都是使用递归方法，先检索根结点的左子树，再检索根，然后检索根节点的右子树，如果符合要求就输出该节点的名称及坐标。这样可以保证是升序输出。  class CityDB  {  private:      BST<string, Location> \*database;  public:      CityDB()      {          database = new BST<string, Location>();      }      ~CityDB()      {          delete database;      }        void insert(int number)      {          int i;          string name;          int lox,loy;          for(int i=0; i<number; i++)          {              cin >> name >> lox >> loy;              database->insert(name,Location(lox,loy));          }      }        void runCmd(string cmd = "cmd")      {          if (cmd == "remove")          { //remove              string strBuffer;              cin >> strBuffer;              database->remove(strBuffer);          }          else if (cmd == "insert")          { //insert              insert(1);          }          else if (cmd == "selectKey")          { //select key              char c;              cin >> c;              printCaptital(database->getroot(),c);            }          else if (cmd == "selectLocation")          {             int p,q,dis;              cin >> p >> q >> dis;              printDistance(database->getroot(),p,q,dis);          }          else if (cmd == "print")          {              database->print();          }            else          {              cerr << "command is unvailable" << cmd << endl;          }      }      void printCaptital(BSTNode<string,Location>\*bst,char c)  //检索首字母      {          if(bst)          {              printCaptital(bst->left(),c);              if((bst->key())[0]==c)                  cout << bst->key() << " " << bst->element().getx() << " " << bst->element().gety() << endl;              printCaptital(bst->right(),c);          }      }     void printDistance(BSTNode<string,Location>\*bst, int p, int q, int dis) //检索距离     {         if(bst)         {             printDistance(bst->left(),p,q,dis);             if((pow(((bst->element()).getx()-p)\*1.0,2\*1.0)+pow(((bst->element()).gety()-q)\*1.0,2\*1.0)) <= dis\*dis\*1.0)                  cout << bst->key() << " " << bst->element().getx() << " " << bst->element().gety() << endl;              printDistance(bst->right(),p,q,dis);         }     }  };   1. main函数实现   main函数建立了一个城市数据库，按题目要求实现插入、删除、检索、输出等操作  int main()  {      int N;      cin >> N;      CityDB mycity;      mycity.insert(N);      int num;      while(cin>>num)      {          if(num==1)          {              mycity.runCmd("insert");          }          else if(num==0)          {              mycity.runCmd("remove");          }          else if(num==2)          {              break;          }        }      mycity.runCmd("print");      mycity.runCmd("selectKey");      mycity.runCmd("selectLocation");      system("pause");      return 0;  } | | | | | | |
| 1. 实验结果及分析和（或）源程序调试过程 2. 实验结果     由图，可以实现实验要求   1. 调试过程   一开始Location类没有实现重载!=，然后发现BST中的!=会报错，于是重载后程序正常。 | | | | | | |

备注：

1、教师在布置需撰写实验报告的实验前，应先将报告书上的“实验题目”、“实验性质”、“实验目的”、“实验项目内容”等项目填写完成，然后再下发给学生。

2、教师在布置需撰写报告的实验项目时，应告知学生提交实验报告的最后期限。

3、学生应按照要求正确地撰写实验报告：

* 1. 在实验报告上正确地填写“实验时间”、“实验地点”等栏目。
  2. 将实验所涉及的源程序文件内容（实验操作步骤或者算法）填写在“实验过程或算法（源程序）”栏目中。
  3. 将实验所涉及源程序调试过程（输入数据和输出结果）或者实验的分析内容填写在“实验结果及分析和（或）源程序调试过程”栏目中。
  4. 在实验报告页脚的“报告创建时间：”处插入完成实验报告时的日期和时间。
  5. 学生将每个实验完成后，按实验要求的文件名通过网络提交（上载）到指定的服务器所规定的共享文件夹中。每个实验一个电子文档，如果实验中有多个电子文档（如源程序或图形等），则用WinRAR压缩成一个压缩包文档提交，压缩包文件名同实验报告文件名（见下条）。
  6. 提交的实验报告电子文档命名为：“年级（两位数字不要“级”字）专业（缩写：计算机科学与技术专业（计科）、网络工程专业（网络）、信息安全专业（信息）、物联网工程（物联网））班级（两位数字）学号（八位数字）姓名实验序号（一位数字）．doc。如学号为20115676、年级为2011级、专业为“计算机科学与技术”专业、班级为“02班”、姓名为“王宇”的学生，完成的第一次实验命名为： 11计科02班20115676王宇1．Doc，以后几次实验的报告名称以此类推。

4、教师（或助教）在评价学生实验时，应根据其提交的其他实验相关资料（例如源程序文件等）对实验报告进行仔细评价。评价后应完成的项目有:

1. 在“成绩”栏中填写实验成绩。每个项目的实验成绩按照总分10评分。
2. 在“教师评价”栏中用符号标注评价项目结果（用√表示正确，用×表示错误，用≈表示 半对半错）。
3. 在“教师评价”栏中“评价教师签名”填写评价教师（或助教）姓名。将评价后的实验报告转换为PDF格式文件归档。
4. 课程实验环节结束后，任课教师将自己教学班的实验报告文件夹进行清理。在提交文件夹中，文件总数为实验次数×教学班学生人数（如，教学班人数为90人，实验项目为5，其文件数为：90×5=450）。任课教师一定要认真清理，总数相符，否则学生该实验项目不能得分。最后将学生提交的实验报告刻光盘连同实验成绩一起放入试卷袋存档。