重庆大学本科学生实验项目任务书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验题目 | 有理数的堆排序 | | | |
| 实验时间 | 2021.12.15 | 实验地点 | | DS1404 |
| 实验性质 | □验证性 √设计性 □综合性 | | | |
| 实验目的  1. 掌握最小堆的基本概念，堆的基本运算以及堆排序的流程  2. 掌握有理数类的定义及逻辑运算规则 | | | | |
| 实验内容：  1. 完成有理数的类定义以及有理数逻辑运算函数  class Rational{  friend bool operator<(const Rational& r1, const Rational& r2) {}  friend bool operator<=(const Rational& r1, const Rational& r2) {}  friend bool operator>(const Rational& r1, const Rational& r2) {}  friend bool operator>=(const Rational& r1, const Rational& r2) {}  friend bool operator==(const Rational& r1, const Rational& r2) {}  friend bool operator!=(const Rational& r1, const Rational& r2) {}  public:  int N; //分子  int D; //分母， 要求大于0    Rational() {} //default constructor  Rational(int n){} //constructor for integer value  Rational(int n, int d) {} //normal constructor  Rational(const Rational& r){} //copy constructor  Rational& operator=(const Rational& r) {} // assignment override  }；  2. 创建有理数的最小堆，实现siftdown, siftup, insert等功能  3． 实现基于最小堆的堆排序，按从小到大的顺序输出有理数  4． 为在线测评系统检测程序的运行，对程序文档及IO做如下规范：  （1）所有类、函数及主程序都写在一个**单cpp文档**里，不能有其他include用的.h或.cpp文档  （2）程序不能输出任何提示用的字符串  （3）输入： 第一行包含一个整数T (1≤T≤105)；接下来的T行，每一行有两个整数n, d (|n|<103, 0<d<103)，用空格隔开，表示输入的有理数的分子和分母。  （4）输出：第一行输出有理数的最小堆序列，第二行输出从小到大排序后的序列。  （5）输出的每个有理数必须规约，以n/d的形式输出，其中d>0且gcd(n,d)=0；如果d=1或n=0则直接输出n  （例子）  输入：  5 //有理数个数  3 2 //第1个有理数 3/2  1 3 //第2个有理数 1/3  4 2 //第3个有理数 4/2  12 10 //第4个有理数 12/10  4 6 //第5个有理数 4/6  输出：  1/32/3 2 6/5 3/2 //第一行 最小堆  1/3 2/3 6/5 3/2 2 //第二行 排序结果 | | | | |
| 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 12月 2 日 | | | 完成日期 2021 年 12 月 15 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、2020。

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** | 有理数的堆排序 | | | | | |
| **实验时间** | **2021.12.15** | | **实验地点** | **DS1404** | | |
| **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： | | | | | | |
| 实验目的  1. 掌握最小堆的基本概念，堆的基本运算以及堆排序的流程  2. 掌握有理数类的定义及逻辑运算规则 | | | | | | |
| 实验内容：  1. 完成有理数的类定义以及有理数逻辑运算函数  class Rational{  friend bool operator<(const Rational& r1, const Rational& r2) {}  friend bool operator<=(const Rational& r1, const Rational& r2) {}  friend bool operator>(const Rational& r1, const Rational& r2) {}  friend bool operator>=(const Rational& r1, const Rational& r2) {}  friend bool operator==(const Rational& r1, const Rational& r2) {}  friend bool operator!=(const Rational& r1, const Rational& r2) {}  public:  int N; //分子  int D; //分母， 要求大于0    Rational() {} //default constructor  Rational(int n){} //constructor for integer value  Rational(int n, int d) {} //normal constructor  Rational(const Rational& r){} //copy constructor  Rational& operator=(const Rational& r) {} // assignment override  }；  2. 创建有理数的最小堆，实现siftdown, siftup, insert等功能  3． 实现基于最小堆的堆排序，按从小到大的顺序输出有理数  4． 为在线测评系统检测程序的运行，对程序文档及IO做如下规范：  （1）所有类、函数及主程序都写在一个**单cpp文档**里，不能有其他include用的.h或.cpp文档  （2）程序不能输出任何提示用的字符串  （3）输入： 第一行包含一个整数T (1≤T≤105)；接下来的T行，每一行有两个整数n, d (|n|<103, 0<d<103)，用空格隔开，表示输入的有理数的分子和分母。  （4）输出：第一行输出有理数的最小堆序列，第二行输出从小到大排序后的序列。  （5）输出的每个有理数必须规约，以n/d的形式输出，其中d>0且gcd(n,d)=0；如果d=1或n=0则直接输出n  （例子）  输入：  5 //有理数个数  3 2 //第1个有理数 3/2  1 3 //第2个有理数 1/3  4 2 //第3个有理数 4/2  12 10 //第4个有理数 12/10  4 6 //第5个有理数 4/6  输出：  1/32/3 2 6/5 3/2 //第一行 最小堆  1/3 2/3 6/5 3/2 2 //第二行 排序结果 | | | | | | |
| 1. 实验过程或算法（源程序） 2. 实现Rational类   Rational类利用gcd()函数（找最大公约数）在初始化的时候对分数化简为最简状态,比较大小的实现则基于将两分数通分后对分子大小的比较。printhelp()函数则实现了按要求输出分数。  int gcd(int a, int b)  {      if(a % b == 0)          return b;      return gcd(b, a%b);  }  class Rational{  friend bool operator<(const Rational& r1, const Rational& r2)  {  return r1.N \* r2.D < r1.D \* r2.N;}  friend bool operator<=(const Rational& r1, const Rational& r2)  {   return r1.N \* r2.D <= r1.D \* r2.N;}  friend bool operator>(const Rational& r1, const Rational& r2)  {   return r1.N \* r2.D > r1.D \* r2.N;}  friend bool operator>=(const Rational& r1, const Rational& r2)  {   return r1.N \* r2.D >= r1.D \* r2.N;}  friend bool operator==(const Rational& r1, const Rational& r2)  {   return r1.N \* r2.D == r1.D \* r2.N;}  friend bool operator!=(const Rational& r1, const Rational& r2)  {   return r1.N \* r2.D != r1.D \* r2.N;}  public:     int N; //分子     int D; //分母， 要求大于0       Rational() {}  //default constructor     Rational(int n){ N=n; D=1;} //constructor for integer value     Rational(int n, int d)     {          if(n==0)          {              N = n;              D = d;          }          else          {              int c = gcd(abs(n),d);              N = n/c;              D = d/c;          }      } //normal constructor     Rational(const Rational& r){N=r.N; D=r.D;} //copy constructor     Rational& operator=(const Rational& r) {N=r.N; D=r.D; return \*this;} // assignment override      void printhelp()      {          if(N==0)              cout << 0;          else if(D==1)              cout << N;          else              cout << N << "/" << D;          return;      }  };   1. 实现最小堆   基于课本定义实现最小堆，包括shiftdown()、构造函数、isLeaf()(判断是否为叶子结点)、leftchild(),rightchild()(返回左、右孩子)、parent()返回父节点)、insert()(插入元素)、removefirst()(删除第一个元素)、remove()(删除指定位置元素)、printhelp()(输出堆元素)、heapsort()(堆排序)。其中堆的元素类型为泛型，可根据用户需要类型传入参数，具有通用性。  template<typename E>  class heap  {  private:      E \* Heap;      int maxsize;      int n;      void shifdown(int pos)  //helper to put element at its correct position      {          while(!isLeaf(pos))          {              int j = leftchild(pos);              int rc = rightchild(pos);              if((rc < n) && (Heap[rc]<Heap[j]))                  j = rc;              if(Heap[pos]<Heap[j])                  return;              swap(Heap,pos,j);              pos = j;          }      }  public:      heap(E \*h, int num, int max)      {Heap = h; n = num; maxsize = max; buildHeap();}        int size() const      {return n;}      bool isLeaf(int pos) const      {return (pos>=n/2)&&(pos<n);}      int leftchild(int pos) const      {return 2\*pos+1;}      int rightchild(int pos) const      {return 2\*pos+2;}      int parent(int pos) const      {return (pos-1)/2;}      void buildHeap()      {          for(int i=n/2-1; i>=0; i--)          shifdown(i);      }      void insert(const E & it)      {          if(n>=maxsize)          {              cout << "Heap is full";              return;          }          int curr = n++;          Heap[curr] = it;          while((curr!=0) && (Heap[curr] < Heap[parent(curr)]))          {              swap(Heap,curr,parent(curr));              curr = parent(curr);          }      }      E removefirst()      {          if(n<=0)          {              cout << "Heap is empty";              return NULL;          }          swap(Heap,0,--n);  //Swap the first with last value          if(n!=0)              shifdown(0);          return Heap[n];      }      E remove(int pos)      {          if(pos<0 || pos>=n)          {              cout << "Bad position";              return NULL;          }          if(pos==(n-1))              n--;          else          {              swap(Heap,pos,--n);              while((pos!=0) && (Heap[pos] < Heap[parent(pos)]))              {                  swap(Heap,pos,parent(pos));                  pos = parent(pos);              }              if(n!=0)                  shifdown(pos);          }          return Heap[n];      }      void printheap() {           //print Heap          for (int i = 0; i < n; i++) {              printhh(Heap[i]);              cout << "  ";          }          return;      }      void heapsort(int n)      {          E maxval;          for(int i=0; i<n; i++)          {              maxval = this->removefirst();          }          return;      }  };   1. 补充函数 2. Swap函数   可实现不同类型元素的交换操作  template<typename E>  void swap(E \* a, int i, int j)  {      E temp = a[i];      a[i] = a[j];      a[j] = temp;      return;  }   1. Printhh()函数   在堆中调用该函数，该函数又调用了泛型E类中的printhelp()函数，从而实现了堆可以根据用户要求按格式输出不同类型的元素，只需要用户实现所需要类型的printhelp()函数即可。  template<typename E>  void printhh(E a)  {      a.printhelp();  }   1. Main()函数   按要求输入数据，并建堆，实现堆排序。  int main()  {      int T;      cin >> T;      Rational compute[10];      for(int i=0; i<T; i++)      {          int son;          int mum;          cin >> son >> mum;          compute[i] = Rational(son,mum);      }        heap<Rational> tryone(compute,T,T);      tryone.printheap();      cout << endl;      tryone.heapsort(T);      for(int i=T-1; i>=0; i--)      {          compute[i].printhelp();          if(i==0)              cout << endl;          else              cout << "  ";      }        system("pause");      return 0;  } | | | | | | |
| 1. 实验结果及分析和（或）源程序调试过程   1.结果分析    由图片可知，该程序实现了实验要求，即对有理数建堆并排序。 | | | | | | |

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20155676、年级为2015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： 15计科02班20155676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。