山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201700140056 | 姓名：李港 | | 班级：跟18.2（实17.4） |
| 实验题目：实验一 递归练习 | | | |
| 实验学时：2h | | 实验日期：2019.09.12 | |
| 实验目的：  1、熟悉开发工具的使用。  2、掌握递归的实现思想。 | | | |
| 软件开发工具：  Virtual Studio 2019 | | | |
| 1. **实验内容**   1、键盘输入n和n个互不相同的整数，输出n个整数的全排列。  2、键盘输入n和n个互不相同的整数，输出n个整数的所有子集。   1. **数据结构与算法描述（整体思路描述，所需要的数据结构与算法）**   1.全排列：  **思路：**实现全排列需要调换各个元素的位置，有两种调换方案：开辟新的内存空间保存置换结果或直接在原内存空间进行置换。为了更好地体现递归中逐层改变条件的思想，此处选择采用第二种方案。递归函数内部在调用递归函数前后均进行位置置换，以修改或恢复当前状态。  **数据结构：**本实验对数据结构要求不高，采用C原生数组  **算法：**采用递归算法  2.输出子集：  **思想：**采用两个数组：数据数组与bool数组，在子集中，每个元素有出现或不出现两种可能，故此处采用一bool数组保存元素的两种状态，在逐层递归过程中也逐层修改bool数组内容，每次递归均进行递归尽头条件判断，条件符合即根据bool数组输出数据数组的内容。  **数据结构：**同上，采用原生数组  **算法：**递归算法   1. **测试结果（测试输入，测试输出）**   1.全排列：  输入：  **3 1 2 3**  输出：  **1 2 3**  **1 3 2**  **2 1 3**  **2 3 1**  **3 2 1**  **3 1 2**  2.求子集  输入：  **3 1 2 3**  输出：  **{}**  **{3}**  **{2}**  **{2,3}**  **{1}**  **{1,3}**  **{1,2}**  **{1,2,3}**   1. **分析与探讨（结果分析，若存在问题，探讨解决问题的途径）**   子集与全排列程序结果均正常，但在代码编写过程中，遇到了以下问题：  若遵循严格的责任界限，则算法类没有承担输入工作，也就不应该承担输出工作，但是输出工作若在外部进行，未免有些繁琐；如果同时承担输入输出工作，那类就承担了过多与算法无关的工作。类似这种编码规范的问题，与算法无关，但还是需要注意。  解决方案：为了提高程序可读性，本实验选择类外输入，类内输出的方式。   1. **附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）**   程序1-1 全排列  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* permutations.cpp \*  \* Copyright (C) 2019 TriAlley lg139@139.com. \*  \* \*  \* @brief 递归求全排列的测试程序 \*  \* @author TriAlley \*  \* @email lg139@139.com \*  \* @version 1.0 \*  \* @license GNU General Public License (GPL) \*  \* \*  \* Change History : \*  \* <Date> | <Version> | <Author> | <Description> \*  \* 2019/09/11 | 1.0 | TriAlley | Create file \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream>  using namespace std;  /\*\*  \* @brief 求全排列类  \* private:  \* T\* arr; //元素数组  \* int num; //元素个数  \* void lgswap(T& a, T& b); //自行编写的交换函数  \* void show(); //递归结束时显示全排列  \*  \* public:  \* void calpermutation(int i=0); //求全排列入口  \*/  template<class T>  class permutation {  private:  T\* arr;  int num;  /\*  \* 功能：显示函数，用于在递归尽头时打印一次结果  \* 输入：两个需要置换的元素  \* 返回：无  \* 特殊说明：iostream中有swap函数，在此使用自己编写的lgswap  \*/  void lgswap(T& a, T& b) {  T temp;  temp = a;  a = b;  b = temp;  }  /\*  \* 功能：显示函数，用于在递归尽头时打印一次结果  \* 输入：无  \* 返回：无  \*/  void show() {  for (int i = 0; i < num; i++) {  cout << arr[i] << " ";  }  cout << endl;  return;  }  public:  permutation(int inum, T\* iarr) :num(inum), arr(iarr) {}  /\*  \* 功能：全排列递归主函数  \* 输入：  \* i：当前递归是第几层递归，也即排列到了第几个变量，默认为0  \* 返回：无  \*/  void calpermutation(int i=0) {  /\*如果是最后一层，则说明这是集合的一个全排列，输出数组内容并返回到上一层\*/  if (i+1 == num) {  show();  }  /\*非最后一层的递归，将当前层与剩余各层进行置换，并进入下一层递归\*/  for (int j = i; j < num; j++) {  lgswap(arr[j], arr[i]);  calpermutation(i + 1);  lgswap(arr[j], arr[i]);  }  }  };  int main() {  /\*获取输入，初始化数组\*/  cout << "请输入数组大小：";  int num;  cin >> num;  cout << "请输入元素内容，空格分开：";  int\* arr = new int[num];  for (int i = 0; i < num; i++) {  cin >> arr[i];  }  /\*构造类，传入数组大小与数组指针，调用全排列函数\*/  permutation<int> p(num,arr);  p.calpermutation();  /\*谁申请谁释放，主函数释放数组\*/  delete[] arr;  return 0;  }  程序1-2 求子集  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* subset.cpp \*  \* Copyright (C) 2019 TriAlley lg139@139.com. \*  \* \*  \* @file subset.cpp \*  \* @brief 递归求子集程序测试 \*  \* @author TriAlley \*  \* @email lg139@139.com \*  \* @version 1.0 \*  \* @license GNU General Public License (GPL) \*  \* \*  \* Change History : \*  \* <Date> | <Version> | <Author> | <Description> \*  \* 2019/09/11 | 1.0 | TriAlley | Create file \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include<iostream> //使用输入输出  #include<cstring> //使用memset  using namespace std;  /\*\*  \* @brief 求子集类  \* private:  \* T\* arr; //元素数组  \* int n; //元素个数  \* bool\* isshowarr; //显隐控制数组  \* void show(); //递归结束时显示集合  \* void isubset(int i, bool isshow); //求子集主体函数  \*  \* public:  \* subset(T\* ia,int in); //构造函数  \* void calsubset(); //求子集入口  \*/  template<class T>  class subset {  private:  T\* arr;  int n;  bool\* isshowarr;  /\*\*  \* @brief 递归返回前显示当前状态  \*/  void show() {  cout << "{";  bool showcam = false;  for (int i = 0; i < n; i++) {  if (isshowarr[i]) {  if (showcam) {  cout << ",";  }  cout << arr[i];  showcam = true;  } else {  }  }  cout << "}" << endl;  }  /\*\*  \* @brief 求子集主体函数  \* int i 当前控制的元素之下标，是递归的结束条件之一  \* bool isshow 指定i所指向元素的显隐  \*/  void isubset(int i, bool isshow) {  isshowarr[i] = isshow;  if (i + 1 == n) {  show();  return;  }  isubset(i + 1, false);  isubset(i + 1, true);  }  public:  /\*\*  \* @brief 构造函数 申请资源  \* int in 元素总数量  \* T\* ia 元素数组指针  \*/  subset(T\* ia, int in) :n(in), arr(ia) {  /\*对元素显隐控制数组进行初始化\*/  isshowarr = new bool[in];  memset(isshowarr, 0, sizeof(bool) \* n);  }  /\*\*  \* @brief 析构函数 释放资源  \*/  ~subset() {  //delete[] arr; //谁申请谁释放，类不申请元素数组的内存，无需在此释放  delete[] isshowarr;  }  /\*\*  \* @brief 子集计算的入口  \*/  void calsubset() {  isubset(0, false);  isubset(0, true);  }  };  int main() {  /\*获取数组大小与内容\*/  int num;  cout << "请输入数组大小：";  cin >> num;  int\* arr = new int[num];  cout << "请输入数组内容，空格分开：";  for (int i = 0; i < num; i++) {  int temp;  cin >> temp;  arr[i] = temp;  }  /\*初始化求子集类并调用函数\*/  subset<int> s(arr, num);  s.calsubset();  /\*谁申请谁释放\*/  delete[] arr;  return 0;  } | | | |