

**实验报告封面**

**课程名称： 算法课程设计 课程代码： GF2001**

**任课老师： 李俊琴 实验指导老师: 李俊琴**

# 实验报告名称： 实验1 排序算法及其应用

学生姓名： 余庆祥

学号： 2220631136 教学班： BIB

递交日期： 2022/12/1

签收人：

我申明，本报告内的实验已按要求完成，报告完全是由我个人完成，并没有抄袭行为。我已经保留了这份实验报告的副本。

申明人(签名):

**实验报告评语与评分：**

**评阅老师签名：**

**内页写作格式**

**一、实验名称：**要用最简练的语言反映实验的内容。

**二、实验日期：**写明做实验的具体年、月、日及组別。

**三、实验目的：**使用简洁的文字或关键字来敘述，是以怎样的目的作此实验的。

**四、实验环境：**实验的操作系统和软件等。

**五、实验的步骤和方法：**这是实验报告极其重要的内容。这部分要写明经过哪几个步骤。

**六、数据记录和计算：**指从实验中测到的数据以及计算结果。

**七、实验结果或结论：**即根据实验过程中所见到的现象和测得的数据，得出结论。

**八、备注或说明**：可写上实验成功或失败的原因，实验后的心得体会、建议等。

|  |  |
| --- | --- |
| **1、实验名称** | **实验1 排序算法及其应用** |
| **2、实验日期** | **2022/12/1** |
| **3、实验目的** | |
| 1. 掌握算法设计概念及流程 2. 掌握算法评价的方法 | |
| **4、实验环境** | |
| 1、硬件环境：i386兼容机，2G剩余硬盘空间；  2、软件环境：Windows XP以上兼容系统；  3、使用资源：开发环境VC++2017。 | |
| **5、实验的步骤和方法** | |
| 1. **给定n个整数组成的序列a1, a2, …, an, 用快速排序法将该序列排序。**  * 说明：快速排序是一种交换排序，它由C. A. R. Hoare在1962年提出。快速排序(quick sort)的采用了分而治之(divide and conquer)的策略：将原问题分解为若干个规模更小但结构与原问题相似的子问题。递归地解这些子问题，然后将这些子问题的解组合为原问题的解。 * 算法思想：快速排序的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分：分割点左边都是比它小的数，右边都是比它大的数。然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。 * 图解：  1. **根据上述理解，设计算法，实现对序列的排序算法**   **注意：首先画出设计算法的流程图，然后再根据流程图进行编程实现。**   1. **描述游戏中的英雄，每英雄有自己的属性：姓名，生命值，防御力，攻击力，魔法值等。编程实现召唤英雄组成战队的功能。根据英雄的某个属性将英雄排序，然后输出该英雄的信息。要求：**  * **控制台程序** * **英雄的存储空间是动态增长的** (使用melloc、realloc函数) * **不能使用stl或其它第三方库** * **使用快速排序法**  1. **拓展练习：**  * **使用堆排序和桶排序两种排序算法实现第（3）步的功能**   完成上面的练习后，填写实验报告，将**实验报告word、运行结果录屏，分别上交智慧树**。 | |
| **6、数据记录和计算** | |
| （快排算法的流程图，1、2题关键代码请贴在此处）  流程图：  代码实现：  #include <iostream>  #include <vector>  using namespace std;  int part(int \*v, int low, int height)  {  int i = low, j = height, key = v[low];  while(i < j)  {  while(i < j && v[j] > key)  {  j--;  }  if(i < j)  {  auto temp = v[i];  v[i] = v[j];  v[j] = temp;  i++;  }  while(i < j && v[i] <= key)  {  i++;  }  if(i < j)  {  auto temp = v[i];  v[i] = v[j];  v[j] = temp;  j--;  }  }  return i;  }  void Quicksort(int \*v, int low, int height)  {  int mid;  if(low < height)  {  mid = part(v, low, height);  Quicksort(v, low, mid - 1);  Quicksort(v, mid + 1, height);  }  }  int length(int \*v)  {  cout << sizeof(v) << endl;  cout << sizeof(&v) << endl;  return 0;  }  int main()  {  int v[] = {3, 2, 9, 1, 4, 0};  int v\_length = sizeof(v) / sizeof(&v);  Quicksort(v, 0, v\_length - 1);  for(int i = 0; i < v\_length; i++)  {  cout << v[i] << endl;  }  return 0;  }  第三题代码：  #include <iostream>  using namespace std;  struct heroAttr {  char const\* name;  int attk;  int def;  int hp;  int mp;  heroAttr(){  this->name = "null";  this->attk = 0;  this->def = 0;  this->hp = 0;  this->mp = 0;  };  heroAttr(char const\* name,  int attk,  int def,  int hp,  int mp){  this->name = name;  this->attk = attk;  this->def = def;  this->hp = hp;  this->mp = mp;  };  };  struct allHero {  heroAttr hero[3];  int len;  };  enum heroAttrType {  attk, def, hp, mp  };  int part(heroAttr \*v, int low, int height, heroAttrType sortType)  {  int i = low, j = height, key;  switch(sortType)  {  case attk:  key = v[low].attk;  break;  case def:  key = v[low].def;  break;  case hp:  key = v[low].hp;  break;  case mp:  key = v[low].mp;  break;  }  while(i < j)  {  while(i < j && v[j].attk > key)  {  j--;  }  if(i < j)  {  auto temp = v[i];  v[i] = v[j];  v[j] = temp;  i++;  }  while(i < j && v[i].attk <= key)  {  i++;  }  if(i < j)  {  auto temp = v[i];  v[i] = v[j];  v[j] = temp;  j--;  }  }  return i;  }  void Quicksort(heroAttr \*v, int low, int height, heroAttrType heroattrtype)  {  int mid;  if(low < height)  {  mid = part(v, low, height, heroattrtype);  Quicksort(v, low, mid - 1, heroattrtype);  Quicksort(v, mid + 1, height, heroattrtype);  }  }  int main()  {  allHero heros;  heros.len = 3;    heros.hero[0] = {"Blue Eyed White Dragon", 3000, 2500, 8000, 4};  heros.hero[1] = {"Black Magician", 2500, 2100, 4000, 5};  heros.hero[2] = {"DDD twin king Cali Yuga", 3500, 3000, 2000, 6};  Quicksort(heros.hero, 0, heros.len, attk);  cout << "升序排序：" << endl;  for(int i = 0; i < heros.len; i++)  {  cout << "No." << i + 1 << ": ";  cout << heros.hero[i].name << endl;  cout << heros.hero[i].attk << " ";  cout << heros.hero[i].def << endl;  cout << heros.hero[i].hp << " ";  cout << heros.hero[i].mp << endl;  }  return 0;  }  效果视频如 01.mp4 所示。 | |
| **7、实验结果或结论**  **）** | |
| **我对“算法”的理解**：  我对算法的理解是：对某些特定的问题建模，并利用合适的数据结构与规律来解决该问题。  **我对排序算法的了解：（种类？区别？应用？）**  **种类有：冒泡排序、快速排序、选择排序等；**  **区别：冒泡与选择排序的空间利用率高，而快速排序的花费时间少；**  **应用：冒泡与选择排序用于空间比较小的环境下，快速排序应用于对速度要求高的环境下。** | |
| **8、备注或说明** | |
| **学会了算法可以帮助开发者选择更适合当前的解决方法。** | |