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：201900161140 | 姓名： 张文浩 | | 班级： 19级人工智能 |
| 实验题目：排序算法 | | | |
| 实验学时：2 | | 实验日期： 9.21 | |
| 实验目的：  掌握各种简单的排序算法 | | | |
| 软件开发工具：  Vscode | | | |
| 1. 实验内容   用任意一种排序方式给出n个整数按升序排序后的结果，满足以下要求：  1.不得使用与实验相关的STL；  2.需使用类模版(template<class T>)；  3.需定义排序类，封装各排序方法；  4.排序数据需使用动态数组存储；  5.排序类需提供以下操作：名次排序、及时终止的选择排序、及时终止的冒泡排序、插入排序。  **输入输出格式**：  **输入**：输入的第一行是一个整数n(1<=n<=1000)，表示需排序的数的个数。接下来一行是n个整数，数的范围是0到1000，每两个相邻数据间用一个空格分隔。  **输出**：一行排好序的序列。  接下来一行是n个整数，数的范围是0到100000，每两个相邻数据间用一个空格分隔。  **输出**：  一个整数，代表所有排列价值的或。   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）   这几种排序算法都可以用简单的数组实现  思路：  名次排序：  先计算每个数组在数组中的排名，存在一个数组r[]，中，然后遍历数组r，将不在正确位置上的数字用swap换到正确的位置上。  即时终止的选择排序：  每次找到前size个数字中最大的数，并与未排序的数字的最后一个位置进行交换。即时终止的条件是在找最大元素时发现后面的一直都比前面的大，说明这个时候已经排序完成，可以直接退出for循环遍历了。  即时终止的冒泡排序：  从头开始不断地两两比较，将大的交换到右边。及时终止的条件是如果在一次冒泡过程中一次交换也没发生，说明已经正确排序，不用再冒泡了。  插入排序：  维护一个有序区，将数据一个一个插入到有序区的适当位置，直到整个数组都有序**。**即每步将一个待排序的记录，按其关键码值的大小插入前面已经排序的文件中适当位置上，直到全部插入完为止。   1. 测试结果（测试输入，测试输出）      1. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）   这次实验中，要注意类的使用。在类中可以创建一些辅助函数用于帮助排序的实现，从而达到规整程序的目的。这次实验所涉及的”及时终止”思想，可以通过布尔类型变量的设定来实现：在循环中判断当前是否需要继续排序，如果不需要则将算法终止，从而减少程序的开销，结合实验一的内容可以看出，布尔类型的数据在编写程序中有着及其广泛且重要的应用。合理使用布尔变量，可使程序更高效。   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）   #include<bits/stdc++.h>  using namespace std;  template <class T>  class mysort  {  public:      mysort(){};      ~mysort(){};  public:      //名次排序、及时终止的选择排序、及时终止的冒泡排序、插入排序。      //名次排序      //计算名次      void rank(T a[], int n, int r[])      {          for (int i = 0; i < n; i++)          {              r[i] = 0;          }          for (int i = 1; i < n; i++)          {              for (int j = 0; j < i; j++)              {                  if (a[j] <= a[i])                      r[i]++;                  else                      r[j]++;              }          }      }      void rearrange(T a[], int n)      {          cout << "rank sort" << endl;          int \*r = new int[n];          rank(a, n, r);          for (int i = 0; i < n; i++)          {              while (r[i] != i)              {                  int t = r[i];                  swap(a[i], a[t]);                  swap(r[i], r[t]);              }          }          delete[] r;      }      //及时终止的选择排序      void selectionsort(T a[], int n)      {          cout << "select sort" << endl;          bool sorted = false;          for (int size = n; !sorted && (size > 1); size--)          {              int indexofmax = 0;              sorted = true;              //查找最大元素              for (int i = 1; i < size; i++)              {                  if (a[indexofmax] <= a[i])                      indexofmax = i;                  else                      sorted = false;              }              swap(a[indexofmax], a[size - 1]);          }      }      //及时终止的冒泡排序      bool bubble(T a[], int n) //把数组a[0;n-1]中的最大的元素移到最右端      {          bool swapped = false; //目前为止为交换          for (int i = 0; i < n - 1; i++)          {              if (a[i] > a[i + 1])              {                  swap(a[i], a[i + 1]);                  swapped = true; //交换              }          }          return swapped;      }      void bubblesort(T a[], int n) //及时终止的冒泡排序      {          cout << "bubble sort" << endl;          for (int i = n; i > 1 && bubble(a, i); i--)              ;      }      //插入排序      void insert(T a[], int n, const T &x) //把x插入有序数组a[0,n-1]      {          int i;          for (i = n - 1; i >= 0 && x < a[i]; i--)          {              a[i + 1] = a[i];          }          a[i + 1] = x;      }      void insertsort(T a[], int n) //对数组a[0:n-1]实施插入排序      {          cout << "insert sort" << endl;          for (int i = 1; i < n; i++)          {              T t = a[i];              insert(a, i, t);          }      }  };  int main()  {      int n;      cin >> n;      int \*a = new int[n];      for (int i = 0; i < n; i++)      {          cin >> a[i];      }      mysort<int> m;      //m.rearrange(a, n);      //m.selectionsort(a, n);      //m.bubblesort(a, n);      m.insertsort(a, n);      for (int i = 0; i < n; i++)      {          cout << a[i] << " ";      }      delete[] a;      system("pause");      return 0;  } | | | |