**\_\_\_\_2019\_\_\_\_\_\_级\_\_\_\_\_信1901-1\_\_\_\_\_班\_\_\_\_\_2020\_\_年\_\_\_\_11\_\_月\_\_27\_日**

**姓名\_\_\_\_\_李朴凡\_\_\_\_\_\_\_ 学号\_\_\_\_\_\_20194311\_\_\_\_\_\_ 电话\_\_\_\_\_18333135009\_\_\_\_\_\_\_\_**

**1.实验题目**

各种内部排序算法的时间复杂度分析结果只给出了算法执行时间的阶，或大概执行时 间。试通过随机的数据比较各算法的关键字比较次数和关键字移动次数，以取得直观感受。 [基本要求]

对以下常用的内部排序算法进行比较：起泡排序、直接插入排序、简单选择排序、快速

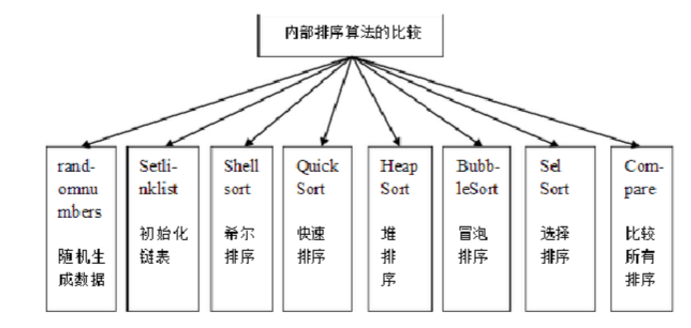
排序、希尔排序、堆排序。

**2.需求分析**

同一组数由不同的方法是进行排序，比较各种算法需要的时间大小和移动次数，对不同算法有直观的体验。

常用排序排序方法有：起泡排序、直接插入排序、简单选择排序、快速排序、希尔排序、堆排序。

**3．概要设计**



**4．详细设计**

数据储存方式

typedef struct

{

int r[MAXSIZE + 1];

int length;

}Sqlist;

//直接插入排序

void InsertSort(Sqlist& L)

//希尔排序

void ShellInsert(Sqlist& L, int dk)

//起泡排序

void BubbleSort(Sqlist &L)

//简单选择排序

void SelectSort(Sqlist& L)

//快速排序

int Partition(Sqlist& L, int low, int high, int &k, int &l)

//堆排序

void HeapAdjust(Sqlist& L, int s, int m, int &k, int &l)

//菜单栏

cout << "请选择：" << endl;

cout << "1 直接插入排序" << endl;

cout << "2 希尔排序 " << endl;

cout << "3 起泡排序" << endl;

cout << "4 简单选择排序" << endl;

cout << "5 快速排序" << endl;

cout << "6 堆排序" << endl;

cout << "7 显示第二组数据" << endl;

cout << "8 显示第三组数据" << endl;

cout << "9 显示第四组数据" << endl;

cout << "10 显示第五组数据" << endl;

cout << "11 退出" << endl;

**5．调试分析**

各种排序的比较次数不同，关键字的移动次数也不尽相同。

**6．使用说明**

按照使用菜单的选项进行点击即可，原数据以设置，不需要人为输入。

**7．测试结果**

