实验一 插入排序算法

1. 用 C/C++语言实现对 n 个整数的直接插入排序，按照升序排序。

要求：按照结构化程序设计思想，分别定义函数 CreateData 和 InsertSort 实现排序功能，然后编写主函数 main 对程序功能进行测试，输出排序结果。

void InsertSort(int a[], int n);

对输入输出数据的进一步要求：

（1） 随机输入一个序列，在界面上输出其排序结果。

（2） 从文件中读取多组测试用例（每一行表示一个序列），输出其排序后的结果，并保存到相应的文件中，每组测试用例输出占一行。（自己设计文件结构）

（3） 由计算机生成若干个 3 位的随机整数，放入数组，完成对数组的升序排序。

提示：调用 rand 库函数产生随机数，随机数的范围在 0~RAND_MAX 之间，需要将其转换到一个更大的范围，可以运用以下四个步骤：

①将 rand 库函数产生的值限制为一个浮点数 d，范围是 0≤d<1。

②用乘法将 d 的值按照需要的范围扩大若干倍；

③将上述值的小数部分截去，即产生以 0 为最小值的一定范围的随机整数；

④修改数值的范围使得从需要的最小值开始。

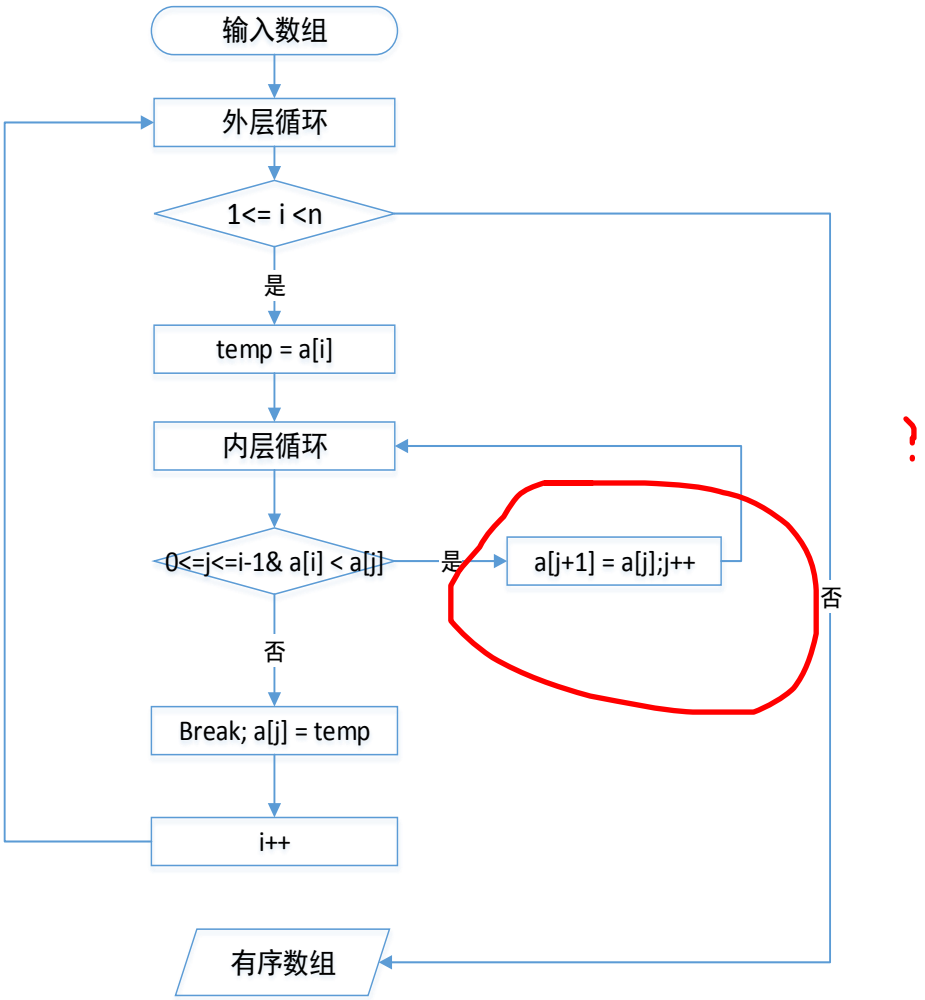
相应地函数：void CreateData(int a[ ],int n,int low,int high)

功能：产生 n 个 low～high 之间的随机整数放入数组 a 中。这样模拟 100 个考试成绩数据，可以调用 CreateData(a，100，0,100)。

关键算法步骤：

1. 从第一个元素开始，该元素可以认为已经被排序；
2. 取出下一个元素，在已经排序的元素序列中从后向前扫描；
3. 如果该元素（已排序）大于新元素，将该元素移到下一位置；
4. 重复步骤 3，直到找到已排序的元素小于或者等于新元素的位置；
5. 将新元素插入到该位置后；
6. 重复步骤 2~5。

流程图如下：



2. 用链表建立一个班级同学录，同学录的信息包括学生姓名、班级、专业、手机号码、QQ 号码、邮箱等。功能要求：

（1） 新建同学录 Creat()；

（2） 在界面上打印所有学生信息 Print(struct Student *)

（3） 打印指定学生的信息 Print(struct Student *, int id)

（4） 插入新加入学生信息 Insert( )

（5） 删除指定学生信息 Delete( );

实验二 归并排序与堆排序

1、用归并排序算法实现对 n 个整数的升序排序。

要求：按照结构化程序设计思想，分别定义函数 CreateData 和 MergeSort 实现排序功能，然后编写主函数 main 对程序功能进行测试，输出排序结果。

输入数据要求：（二选一，建议选择二）

（1）随机输入一个序列，在界面上输出其排序结果。

（2）从文件（如 1.in）中读取多组测试用例，屏幕上显示其排序后的结果，并将结果保存到相应的输出文件（如 1.out）中，每组测试用例输出占一行。（自己设计文件结构）

以下是文件组织结构的一个样例：

样例输入（文件 1.in）

5　　//表示需要排序的数字序列个数

5 3 4 2 1　　//排序序列

4

4 2 3 1

0　　//　0 表示结束

样例输出（文件 1.out）

1 2 3 4 5

1 2 3 4

2、用最小堆实现最小优先队列，实现以下五种操作：

（1）Build(S);　　//将数组 S[1…n]创建成最小队列

（2）Insert(S, x);　　//把元素 x 插入到集合 S 中

（3）Minimum(S);　　//返回 S 中具有最小键字的元素

（4）ExtractMin(S);　//去掉并返回 S 中的具有最小键字的元素

（5）DecreaseKey(S, x, k);　//将元素 x 的关键字值减少 k，使得关键字值变为 x-k

要求：按照结构化程序设计思想，分别定义函数相应函数实现相应功能，然后编写主函数 main 对程序功能进行测试，屏幕上输出结果。以下是集合样例：

S=<15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1>

实验三 快速排序与 3SUM 问题

1. 用 C/C++语言实现快速排序，按照升序排序。

输入描述

从 a.in 文件中读取输入数据，输入数据包含多组测试用例 （如实在不会对文件的操作，可以界面输入输出。后面要求相同）

输出描述

输出排序后的结果，程序结果保存到 a.out 中，每组测试用例输出占一行。

样例输入

以下是 a.in 文件的一个样例：每组测试用例的第一行是一个整数 N，代表该 组测试用例需要排序的数值个数，接下来的一行有 N 个实数 $a_i(1 \le i \le N)$，代表 待排序的数字。当 N=0 结束输入。$0 \le N \le 10^5$， $0 \le a_i \le 10^9$

2

1 9

5

1 2 5 4 3

样例输出

1 9

1 2 3 4 5

2. 用 C/C++语言实现 3Sum 求和问题，并比较不同方法的实际运行时间。

The following problem is called 3SUM and is very famous in algorithm design.

Given 3 arrays of integers of length n=5000 each, A[4999], B[4999], C[4999], how can we find $a \in A$, $b \in B$, $c \in C$ such that a + b = c? (We could also use real numbers, then we need to test $(a + b) - \varepsilon \le c \le (a + b) + \varepsilon$.)

It is not difficult to write a program to solve this problem in $O(n^3)$ time. But can you obtain an $O(n^2)$ time solution? You might want to initiate A, B, C with random integers, say, with the following loop

```
for (i = 0; i< n-1; i++) {
```

```
        A[i] = rand()%n;
        B[i] = rand()%n;
        C[i] = rand()%(7*n);
    }
```

Again, compare the actual running time of these two programs. If the difference is not obvious, increase n until the difference becomes noticeable.

实验四 动态规划

1. 用 C/C++语言实现使用动态规划算法解决最长公共子序列问题：给定两个序列 X={x1,x2,…,xm}和 Y={y1,y2,…,yn}，输出 X 和 Y 的最长公共子序列的长度，基于实际蛋白质数据的测试。完成计算最长公共子序列的函数：

int lcs(char str1[], char str2[])。

输入描述

从 in.txt 文件中读取输入数据，输入数据包含多组测试用例，每组测试用例的占两行，分别为两个序列 X 和 Y, 两个测试用例之间有一空行。

输出描述

对于每个测试样例，输出最长公共子序列长度，并换行。

样例输入

AGCTAG

ACTCC

ATCGTTGAAT

ACGTGATA

样例输出

3

7

1. 给定 K 个整数的序列{ N1, N2, ..., NK }，其任意连续子序列可表示为{ Ni, Ni+1, ..., Nj }，其中 1 <= i <= j <= K。最大连续子序列是所有连续子序列中元素和最大的一个， 例如给定序列{ -2, 11, -4, 13, -5, -2 }，其最大连续子序列为{ 11, -4, 13 }，最大和为 20。 完成计算最大连续子序列之和的函数：

int MaxSubsequenceSum(int A[],int N);

输入描述

从 in.txt 文件中读取输入数据，测试输入包含若干测试用例，每个测试用例占 2 行，第 1 行给出正整数 K(< 10000)，第 2 行给出 K 个整数，中间用空格分

隔。当 K 为 0 时，输入结束，该用例不被处理。

输出描述

对于每个测试样例，输出最大连续子序列的和，并换行。

样例输入

```
6
-2 11 -4 13 -5 -2
10
-10 1 2 3 4 -5 -23 3 7 -21
0
```

样例输出

```
20
10
```

实验五 线性时间的选择算法

1．实现线性时间的选择算法

1）随机化的划分操作

相比较于 QuickSort 中的 PARTITION 操作，随机选择了主元而不是始终选择最后一个，实现时有多种方法，这里提供一种相对简单的思路:

i) 产生一个随机数 pivot（范围 left 到 right）

ii) 将数组的主元 A[pivot]与 A[right]交换。

iii) 执行之前的 PARTITION 操作。

2）完成 randomizedPartition 函数与 randomizedSelection 函数。由于使用了随机函数，需要先在主函数中对随机函数设定种子（srand）.

3）以<13, 19, 9, 5, 12, 8, 7,4,21, 2, 6, 11>为输入，依次输出该数组的第 1，2，3，4..12 个元素。

4）在上一问题中，在调用 randomizedPartition 处加入断点，观察数组的前后变化，记录数组执行 randomizedPartition 的结果(与第 4 次作业类似)，写入实验报告中。另外还需要记录每次的 p(left),q, r(right), i(id)值，并解释一次 p,q,r,i 值的变化的原因，写入实验报告中。

2. 将使用该算法选择中位数与先排序，后选择中位数(sort(A), return A[mid])作比较。改变数据规模，比较两者的运行时间

1）以之前的 CreatData 函数为例，生成多个随机数。

2）增大数据规模，比较两者的运算时间并写入报告。

3）注意事项：

a) 由于两种算法都是对数据进行原址操作，执行后的数组内容均于执行前不相同，为保证输入一致，需要创建具有相同内容的两个数组作为两种算法的输入，而不类使用同一个数组。

b) 整个文件，只用执行一次 srand()即可。

实验六 最小生成树 Kruskal 算法

1. 实现 MST-Kruskal 算法。

输入格式如下:

<number of vertices>

<number of edges>

<endpoint 1> <endpoint 2> <weight>

<endpoint 1> <endpoint 2> <weight>

...

样例输入

5

6

0 3 8

1 2 20

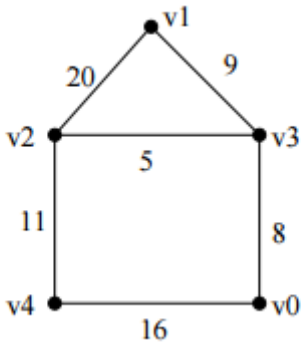
2 3 5

1 3 9

2 4 11

0 4 16

它表示了树:



其最小生成树为：

(v2,v3), (v1,v3), (v0,v3), (v2,v4)

总共的权重为：33.

实验提示

1. 使用 Union Find ADT 数据结构，完成 Make-SET,Union, Find-Set 操作
2. 一条边可以定义为一个含两个结点，一个权重的结构体

```
Struct Edge{
    Int u, v;
    Int weight;
};
```

整个图可以用一个数组表示：Edge graph[number of edges]表示。

3. 所得的 MST(伪代码中的 A)也可用上面的方法表示

实验思考

1）伪代码中的第 5~8 行可以改进， 请问你有什么改进的思路。

2）在 Union 操作处加入断点，观察算法是如何选择边的。

3）有兴趣的同学可以实现 Prime 算法，如果完成了该题，请务必写入实验报告中。