

第十一次作业实验报告

实验环境

以下所有实验都处于这一环境中

操作系统

Windows 10 家庭中文版 64位 版本10.0.17134.345

硬件

CPU Intel Core i7-8750H

RAM 8GB

IDE

Microsoft Visual Studio Community 2017 VisualStudio.15.Release/15.8.5+28010.2036

Visual C++ 2017 00369-60000-00001-AA380

第一题：归并排序

实验目的

编写程序，将两个长度各为10的整型数组数据按从小到大排序，然后将两组数据合并到一个长度为20的整型数组中，合并后的数组仍然按照从小到大排序。

要求：假设每个数组中的元素均无重复；两个小数组的排序算法可以任选；不能使用先合并，然后再次排序的方法，要求一次性同时完成数据合并和排序工作；使用指针变量编程。**说明：自行在main函数中设计测试样例并给出结果截图，要求在实验报告中给出算法思路。**

练习使用指针进行获取变量的值并操作（包括排序和复制等）

实验内容

分析

对两个长度为10的数组进行排序（冒泡、选择、快排等）后，从两个数组的第一位开始，逐个选择较小的赋值给合并后的长度为20的数组即可。

一个简化的例子如下：

```
1 排序好的两个长度为2的数组:
2  [1,3]  [2,4]
3  先比较第一位, 1较小, 赋值给合并后的数组
4  [1,,,]
5  再比较3和2, 2较小
6  [1,2,,]
7  不断重复
8  ...
9  [1,2,3,4]
```

只需让指针指向所需比较的位数获取数值即可。

而对于排序, 题目本身是归并排序的最后一步, 可以构造归并排序来对两个小数组进行操作。也可以直接采用快速排序等。

题目要求自行构造排序样例, 为避免自行构造数组对复杂度的影响, 故选用 `rand()` 进行数组的构造并打印在屏幕上以供检查。

代码

```
1  #include <iostream>
2  using namespace std;
3
4  void arraySort(int *p, int start, int end);
5  void combine(int *p1, int *p2, int *pcombine, int length1, int length2);
6
7  int main() {
8      int a[10], b[10], c[20];
9      cout << "排序样例为: \n";
10     //用随机函数生成样例并打印在屏幕上
11     for (int i = 0; i < 20; ++i) {
12         if (i < 10) {
13             a[i] = rand();
14             cout << a[i] << " ";
15             if (i == 9) cout << endl;
16         }
17         else {
18             b[i - 10] = rand();
19             cout << b[i - 10] << " ";
20         }
21     }
22     cout << endl;
23     int *p1 = &a[0], *p2 = &b[0], *pc = &c[0];
24     //对两个样例进行排序操作
25     arraySort(p1, 0, 9);
26     arraySort(p2, 0, 9);
27     //数据合并并排序
28     combine(p1, p2, pc, 10, 10);
29     //打印合并后数据
```

```

30     cout << "排序结果为: \n";
31     for (int i = 0; i < 20; ++i) {
32         cout << *(pc + i) << " ";
33     }
34
35     return 0;
36 }
37
38 void arraySort(int *p, int start, int end) {
39     //快速排序
40     //检查排序区间是否正确
41     if (start >= end) return;
42     int i = start, j = end, temp = *(p + i);
43     while (i < j) {
44         //寻找比关键值小的
45         while (i < j && *(p + j) >= temp) {
46             --j;
47         }
48         *(p + i) = *(p + j);
49         //寻找比关键值大的
50         while (i < j && *(p + i) <= temp) {
51             ++i;
52         }
53         *(p + j) = *(p + i);
54     }
55     *(p + i) = temp;
56     //让关键值两侧分别是小的和大的
57     arraySort(p, start, i - 1);
58     arraySort(p, i + 1, end);
59 }
60
61 void combine(int *p1, int *p2, int *pcombine, int length1, int length2) {
62     for (int i = 0, j = 0, k = 0; i < length1 + length2; ++i) {
63         while (j <= length1 && k <= length2) {
64             //选取两数组中较小的放入合并后的数组中
65             //验证两数组是否有已经输出结束的
66             if (k == length2) {
67                 *(pcombine + i) = *(p1 + j);
68                 ++j;
69                 break;
70             }
71             else if (j == length1) {
72                 *(pcombine + i) = *(p2 + k);
73                 ++k;
74                 break;
75             }
76             //比较大小并赋值
77             else if (*(p1 + j) < *(p2 + k)) {
78                 *(pcombine + i) = *(p1 + j);
79                 ++j;
80                 break;
81             }
82             else if (*(p1 + j) >= *(p2 + k)) {

```

```

83         *(pcombine + i) = *(p2 + k);
84         ++k;
85         break;
86     }
87 }
88 }
89 }

```

结果

```

Microsoft Visual Studio 调试控制台
排序样例为：
41 18467 6334 26500 19169 15724 11478 29358 26962 24464
5705 28145 23281 16827 9961 491 2995 11942 4827 5436
排序结果为：
41 491 2995 4827 5436 5705 6334 9961 11478 11942 15724 16827 18467 19169 23281 24464 26500 26962 28145 29358
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 22260) 已退出，返回代码为：0。
按任意键关闭此窗口...

```

分析总结

仅利用指针变量，并不能凸显指针变量的优势，形式上与数组差别不大。查阅资料，若排序中仅交换指针存储的地址可以节省时间，推断本程序还有改进空间。

第二题：约瑟夫问题

实验目的

有 n ($n < 50$) 个人围成一圈，顺序编号。从第 1 个人开始报数 (1, 2, 3)，报数为 3 者退出圈子。问最后留下来的人刚开始时排在几号？

要求：要求使用指针变量来写程序；分为 2 个函数来实现，主函数是输入 n 的值和设置一个保存每人编号数组 `num[50]`；另定义一个 `del(int *p, int n)` 函数来处理退出和打印最后一个编号。`main()` 调用 `del()` 时，调用形式为 `del(num, n)`。做好关键语句的注释。说明： n 个人的编号从 0 开始计数。

利用指针替换数组进行约瑟夫问题的模拟

实验内容

分析

可以采用指针模拟自杀过程，方法类同第八次作业中的分析：

对于此题而言，遍历是比较好想。考虑一个有四十个元素的数组，初始状态全部为 0，表示尚未死亡。从第一个人（编号 0）开始，每数 3 个 0，将第三个 0 改为 1，表示死亡。不断重复这个操作直到只剩一个 0，对应的编号即为约瑟夫应该站的位置。

下面给出一个六个人的时候的示例：

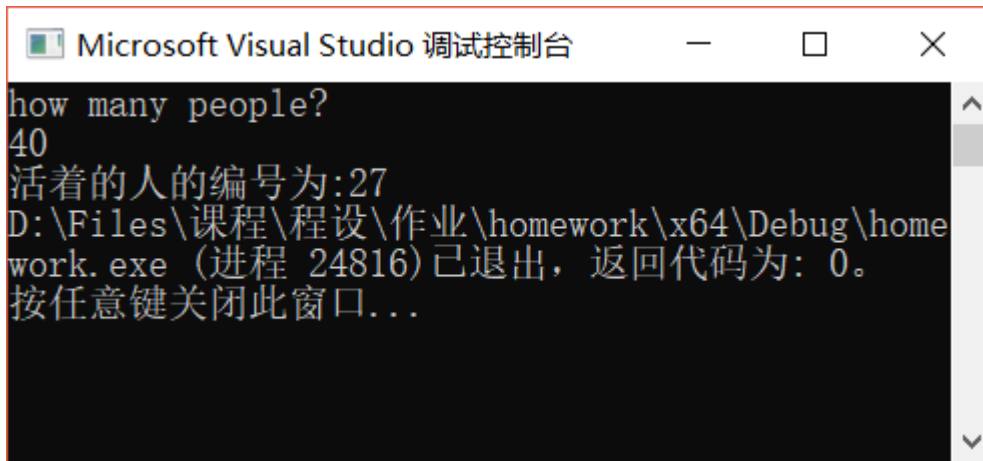
```
1 [0,0,0,0,0,0]
2 [0,0,1,0,0,0]
3 [0,0,1,0,0,1]
4 ...
5 [0,1,1,1,1,1]
```

仅需将数组改为指针即可模拟约瑟夫问题。

代码

```
1 #include <iostream>
2 using namespace std;
3
4 int del(int *p, int n);
5
6 int main() {
7     int n, num[50] = { 0 }; //未死记为0, 死记为1
8
9     cout << "how many people?\n";
10    cin >> n;
11    if (n > 50) {
12        cout << "Error";
13        return -1;
14    }
15
16    return del(num, n);
17 }
18
19 int del(int *p, int n) {
20     int count = 0, i = 0; //count记录报数, i为编号
21     while (count <= 3 * n - 3) { //一共杀掉n-1人, count最多记到3n-3
22         i %= n;
23         if (*(p + i) == 0) {
24             ++count;
25             if (count % 3 == 0) {
26                 *(p + i) = 1; //将这个人从活置为死
27             }
28         }
29         ++i;
30     }
31     for (int i = 0; i < n; ++i) {
32         if (*(p + i) == 0) {
33             cout << "活着的人的编号为:" << i;
34             return 0;
35         }
36     }
37     return -1;
38 }
```

结果



分析总结

同样，这种运用指针的方法和数组高度相似。

如果用 `num[i]` 存储的是第 `i` 个人的编号 `j`，那么模拟约瑟夫问题便复杂为“找0号人，找1号.....找 `i` 号，找还活着的在 `i` 后面的编号最小的人.....”。不过这样的问题可以对应到找顺序编号的人群中活着的，再通过建立一个双射找出他/她是谁。

第三题：数组排序

实验目的

编写程序，从键盘读入10个整数，将其存在一个长度为10的一维数组 `a[]` 中。然后输出该组数据从小到大的排序结果以及在原数组中的下标。输入输出示例：

```
1 input:
2 26 14 57 33 41 12 96 8 67 3
3 output:
4 3 8 12 14 26 33 41 57 67 96
5 9 7 5 1 0 3 4 2 8 6
```

要求：排序算法自选；假设输入数据中无重复数据；程序过程中不能改写数组 `a` 的内容，也不能新开辟整数组存放排序后的结果。

提示：可以开辟一个整型指针的数组：`int * pa[10]`；将原来针对数组 `a[10]` 的排序过程，改为针对数组 `pa[10]` 进行；最后按照 `pa[10]` 的结果打印排序结果和在原来数组 `a[10]` 中的位置。说明：自行在 `main` 函数中设计测试样例并给出结果截图，要求在实验报告中给出算法思路。

要求：排序算法自选；假设输入数据中无重复数据；3/4 程序过程中不能改写数组 `a` 的内容，也不能新开辟整数组存放排序后的结果。提示：可以开辟一个整型指针的数组：`int * pa[10]`；将原来针对数组 `a[10]` 的排序过程，改为针对数组 `pa[10]` 进行；最后按照 `pa[10]` 的结果打印排序结果和在原来数组 `a[10]` 中的位置。说明：自行在 `main` 函数中设计测试样例并给出结果截图，要求在实验报告中给出算法思路。

练习使用指针数组

实验内容

分析

建立一个指针数组，初始化为长度10数组每一元素的地址。

通过对整形数组的“排序”操作，改变指针数组存储地址的顺序。

按照顺序输出指针数组存储的地址中保存的数值，以及这些地址和数组起始地址的差值（即整形数组下标）。

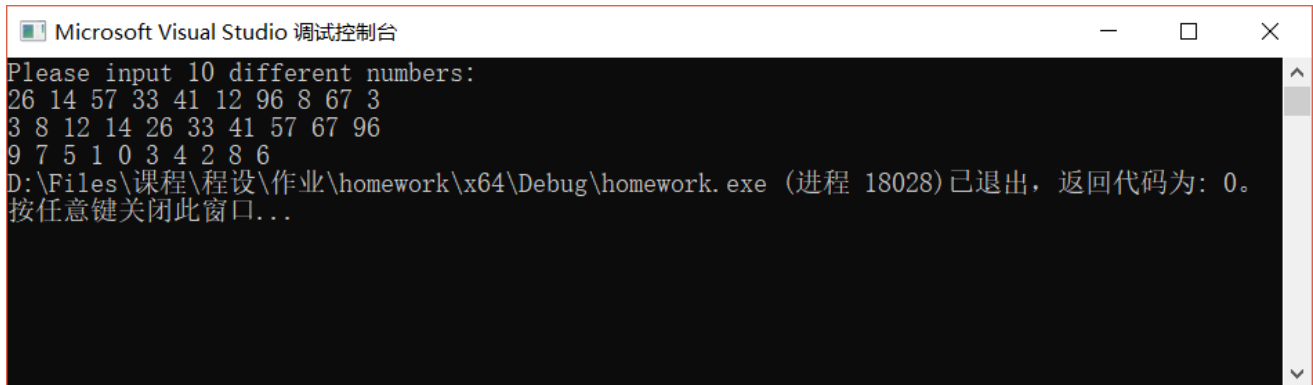
（为便于测试不同测试样例是否能正常实现排序，采用输入的方式获取样例。测试采用样例：26 14 57 33 41 12 96 8 67 3 和 38 23 36 19 68 14 87 70 12 62

代码

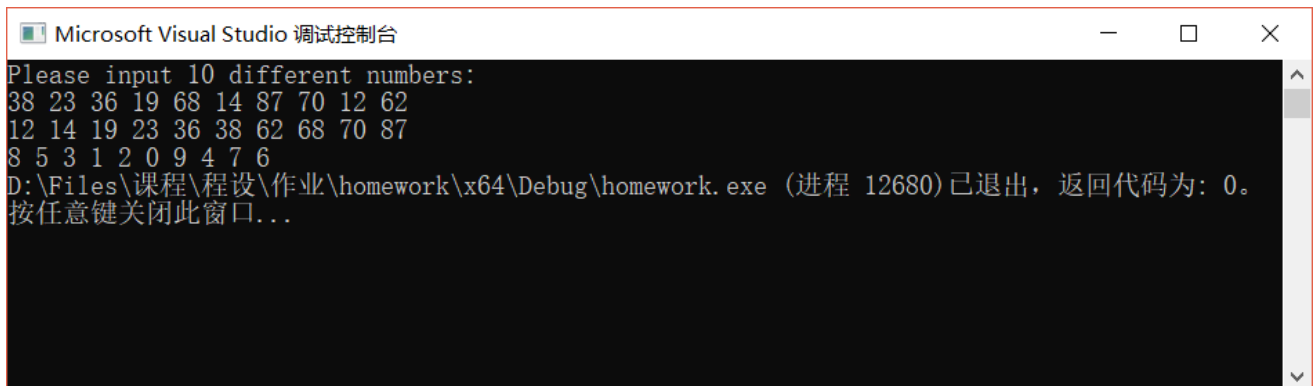
```
1  #include <iostream>
2  using namespace std;
3
4  void arraySort2(int *p[], int start, int end);
5
6  int main() {
7      int a[10];
8      cout << "Please input 10 different numbers:\n";
9      int *pa[10];
10     for (int i = 0; i < 10; ++i) {
11         cin >> a[i];
12         pa[i] = &a[i];
13     }
14     int *p = pa[0]; //记录数组的起始地址
15     arraySort2(pa, 0, 9); //对指针数组进行排序
16     for (int i = 0; i < 10; ++i) {
17         cout << *pa[i] << " "; //输出数组中的值
18     }
19     cout << endl;
20     for (int i = 0; i < 10; ++i) {
21         cout << pa[i]-p << " "; //输出下标
22     }
23
24 }
25
26 void arraySort2(int *p[], int start, int end) {
27     //快速排序
28     //检查排序区间是否正确
29     if (start >= end) return;
30     int i = start, j = end, *temp = p[i];
31     while (i < j) {
32         //寻找比关键值小的
33         while (i < j && *p[j] >= *temp) {
34             --j;
35         }
36         p[i] = p[j];
37         //寻找比关键值大的
38         while (i < j && *p[i] <= *temp) {
39             ++i;
```

```
40     }
41     p[j] = p[i];
42 }
43 p[i] = temp;
44 //让关键值两侧分别是小的和大的
45 arraySort2(p, start, i - 1);
46 arraySort2(p, i + 1, end);
47 }
```

结果



```
Microsoft Visual Studio 调试控制台
Please input 10 different numbers:
26 14 57 33 41 12 96 8 67 3
3 8 12 14 26 33 41 57 67 96
9 7 5 1 0 3 4 2 8 6
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 18028) 已退出，返回代码为：0。
按任意键关闭此窗口...
```



```
Microsoft Visual Studio 调试控制台
Please input 10 different numbers:
38 23 36 19 68 14 87 70 12 62
12 14 19 23 36 38 62 68 70 87
8 5 3 1 2 0 9 4 7 6
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 12680) 已退出，返回代码为：0。
按任意键关闭此窗口...
```

分析总结

交换指针数组存储的地址不改变原数组情况，且通过地址的运算可以获取下标