上机实验报告1

姓名：王越洋 学号：22009200894

**1.1实验一**

（1）题目：求2个或3个正整数中的最大数，用带有默认参数的函数实现。

（2）代码：

1. #include <iostream>
2. using namespace std;
3. *//求2个或3个正整数中的最大数，用带有默认参数的函数实现。*
4. */\**
5. 2个数测试样例：
6. 2
7. 6 1
8. 3个数测试样例
9. 3
10. 0 9 2
11. \*/
12. int maxx(int a = 3, int b = 4, int c = 7) {
13. int max = a;
14. if(b > max) max = b;
15. if(c > max) max = c;
16. return max;
17. }
18. int main()
19. {
20. int a,b,c,n,m;
21. cin >> n;
22. if(n == 2){
23. cin >> a >> b;
24. cout << "输入两个整数：" << a << ' ' << b << endl;
25. m = maxx(a,b);
26. }
27. else{
28. cin >> a >> b >> c;
29. cout << "输入三个整数：" << a << ' ' << b << ' ' << c << endl;
30. m = maxx(a,b,c);
31. }
33. cout << "最大值为：" << m;
35. return 0;
36. }

（3）分析：

1.默认参数：a、b、c 的默认值分别为 3、4 和 7。当调用 maxx 函数时，如果没有传递某些参数，则会使用这些默认值。

2.最大值计算：函数假设 a 是最大值，然后依次与 b 和 c 比较，更新 max 以存储当前最大值。最后返回最大值。

3.默认参数的作用：在主程序中，调用 maxx(a, b) 时，默认会传入 a 和 b，而 c 则使用默认值 7。对于两个数的情况，maxx 函数能够处理，因为 c 的默认值不影响结果。

4.输入部分：读取一个整数 n 来确定输入的数目。如果 n == 2，则输入两个数 a 和 b；如果 n == 3，则输入三个数 a、b 和 c。

5.函数调用：

当 n == 2 时，调用 maxx(a, b)，此时第三个参数使用默认值 7。

当 n == 3 时，调用 maxx(a, b, c)，传入所有三个参数。

6.输出部分：最后程序输出最大值 m。

**1.2实验二**

（1）题目：输入三个整数，将它们按由小到大的顺序输出，要求使用变量的引用。

（2）代码：

1. #include <iostream>
2. using namespace std;
3. *//输入三个整数，将它们按由小到大的顺序输出，要求使用变量的引用。*
4. */\**
5. 测试用例
6. 4 2 19
7. \*/
8. void swap(int &a, int &b) {
9. int temp = a;
10. a = b;
11. b = temp;
12. }
13. void sortt(int &a, int &b, int &c) {
14. if (a > b) swap(a, b);
15. if (a > c) swap(a, c);
16. if (b > c) swap(b, c);
17. }
18. int main() {
19. int x, y, z;
20. cin >> x >> y >> z;
21. sortt(x, y, z);
23. cout << "按从小到大的顺序输出：" << x << " " << y << " " << z << endl;
25. return 0;
26. }

（3）分析：

1.定义swap函数，用于交换两个整数的值。

2.排序和交换函数均使用了引用（&），a和b是在调用swap时传入变量的别名，修改a或b的值将直接修改原始变量的值，而不是其副本，避免了按值传递的缺点，总的时间复杂度为 O(1)。

**1.3实验三**

（1）题目：编写一个程序，用同一个函数名对几个数据进行从小到大排序，数据类型可以是整型、浮点型。用重载函数实现。

（2）代码：

1. #include <iostream>
2. using namespace std;
3. *//编写一个程序，用同一个函数名对几个数据进行从小到大排序，数据类型可以是整型、浮点型。用重载函数实现。*
4. */\**
5. 3
6. 9 3 15
7. 4
8. 2.32 4.51 10.999 3.7
9. \*/
10. const int N = 100010;
11. int main( )
12. {
13. void sort(int a[], int n);
14. void sort(float a[], int n);
15. int a[N];
16. float b[N];
18. int n1,n2;
19. cin >> n1;
20. for(int i = 0;i < n1;i++){*//输入整数*
21. cin >> a[i];
22. }
23. cin >> n2;
24. for(int i = 0;i < n2;i++){*//输入浮点数*
25. cin >> b[i];
26. }
27. sort(a, n1);
28. sort(b, n2);
29. return 0;
30. }
31. *// 整型数组排序*
32. void sort(int a[], int n) {
33. for (int i = 0; i < n - 1; i++) {
34. for (int j = 0; j < n - 1 - i; j++) {
35. if (a[j] > a[j + 1]) {
36. int temp = a[j];
37. a[j] = a[j + 1];
38. a[j + 1] = temp;
39. }
40. }
41. }
42. cout << "整型排序结果：";
43. for (int i = 0; i < n; i++) {
44. cout << a[i] << " ";
45. }
46. cout << endl;
47. }
48. *// 浮点型数组排序*
49. void sort(float a[], int n) {
50. for (int i = 0; i < n - 1; i++) {
51. for (int j = 0; j < n - 1 - i; j++) {
52. if (a[j] > a[j + 1]) {
53. float temp = a[j];
54. a[j] = a[j + 1];
55. a[j + 1] = temp;
56. }
57. }
58. }
59. cout << "浮点型排序结果：";
60. for (int i = 0; i < n; i++) {
61. cout << a[i] << " ";
62. }
63. cout << endl;
64. }

（3）分析：

1.输入声明与存储：

定义了两个数组：a[N] 和 b[N]，分别用来存储整数和浮点数。N 是预定义的最大数组大小 (100010)。

通过两个变量 n1 和 n2 来分别记录输入的整数个数和浮点数个数。程序依次输入整数和浮点数，存储到 a[] 和 b[] 数组中。

2.函数调用：

输入完成后，调用 sort(a, n1) 对整数数组 a 进行排序。

接着调用 sort(b, n2) 对浮点数数组 b 进行排序。

使用函数重载，用简单的冒泡排序重载，同一个 sort 函数名在处理不同类型的数据时调用相应的版本。

**1.4实验四**

（1）题目：对第3题改用函数模板实现，并与第3题程序进行对比分析。

（2）代码：

1. #include <iostream>
2. using namespace std;
3. *//对第3题改用函数模板实现，并与第3题程序进行对比分析。*
4. */\**
5. 3
6. 9 3 15
7. 4
8. 2.32 4.51 10.999 3.7
9. \*/
10. const int N = 100010;
11. template <typename T>
12. void sort(T a[], int n) {
13. for (int i = 0; i < n - 1; i++) {
14. for (int j = 0; j < n - 1 - i; j++) {
15. if (a[j] > a[j + 1]) {
16. T temp = a[j];
17. a[j] = a[j + 1];
18. a[j + 1] = temp;
19. }
20. }
21. }
22. cout << "排序结果：";
23. for (int i = 0; i < n; i++) {
24. cout << a[i] << " ";
25. }
26. cout << endl;
27. }
28. int main() {
29. int a[N];
30. float b[N];
32. int n1, n2;
34. cin >> n1;
35. for (int i = 0; i < n1; i++) {*//输入整数*
36. cin >> a[i];
37. }
39. cin >> n2;
40. for (int i = 0; i < n2; i++) {*//输入浮点数*
41. cin >> b[i];
42. }
44. sort(a, n1);
45. sort(b, n2);
47. return 0;
48. }

（3）分析：

使用函数模板template <typename T>来代替第3题中的函数重载，使得同一个 sort 函数可以处理不同数据类型的排序（如整型和浮点型）。

1.函数模板分析：

template <typename T> 声明了一个模板，T可以代表任意的数据类型。编译器根据传递的参数类型自动生成相应的函数版本。

sort 函数使用了模板 T，因此可以接受任意类型的数组作为参数，例如 int 类型或 float 类型数组。

使用是冒泡排序，通过多次比较和交换相邻元素，将较大的元素逐步移至数组的后。排序的时间复杂度依旧是 O(n²)。

2.优点：

通过模板函数，可以适应多种数据类型，而无需为每种类型单独编写 sort 函数。增强了简洁性、可扩展性。

模板在编译时生成对应类型的具体函数，效率与直接编写针对具体类型的函数几乎相同。