# 5 数组实验

## 5.1 实验目的

（1）掌握数组的说明、初始化和使用。

（2）掌握一维数组作为函数参数时实参和形参的用法。

（3）掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。

（4）掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

## 5.2 实验内容

**5.2.1 源程序改错与跟踪调试**

在下面所给的源程序中，函数strcate(t,s)的功能是将字符串s连接到字符串t的尾部；函数strdelc(s,c)的功能是从字符串s中删除所有与给定字符c相同的字符，程序应该能够输出如下结果：

Programming Language

ProgrammingLanguage Language

ProgrmingLnguge

跟踪和分析源程序中存在的问题，排除程序中的各种逻辑错误，使之能够输出正确的结果。

1. 单步执行源程序。进跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。
2. 跟踪进入函数strdelc时，观察字符数组s中的内容和字符c的值，分析结果是否正确。单步执行for语句过程中，观察字符数组s, j和k值的变化，分析该结果是否存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为何值？分析是否实现了所要求的删除操作。

/\*实验5-1程序改错与跟踪调试题程序\*/

1. #include<stdio.h>
2. void strcate(char [],char []);
3. void strdelc(char [],char );
4. int main(void)
5. {
6. char a[]="Language", b[]="Programming";
7. printf("%s %s\n", b,a);
8. strcate(b,a); printf("%s %s\n",b,a);
9. strdelc(b, 'a'); printf("%s\n",b);
10. return 0;
11. }
12. void strcate(char t[],char s[])
13. {
14. int i = 0, j = 0;
15. while(t[i++]) ;
16. while((t[i++] = s[j++] )!= '\0');
17. }
18. void strdelc(char s[], char c)
19. {
20. int j,k;
21. for(j=k=0; s[j] != '\0'; j++)
22. if(s[j] != c) s[k++] = s[j];
23. }

**解答：**

1. 程序改错：
2. 第五行的b[]="Programming"，需要给b字符串增长空间至可以容纳a字符串，正确形式为：

b[30]="Programming"

1. 第16行需要让计数变量i退回到’\0’的位置，正确形式为：

i--;

1. 第25行，需要为数组的最后一个元素增加终止符，正确形式为：

s[k] = ‘\0’;

1. 修改完上述错误后单步执行源程序：
2. 当单步执行光条刚落在第二个while语句所在行时，i为11，t[i]为0，与预期相符；当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为ProgrammingLanguage和Language，实现了字符串连接。
3. 跟踪进入函数strdelc时，字符数组s中的内容为Language，字符c的值为a，结果正确；字符数组s, j和k值的变化不存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为ProgrmmingLnguge，实现了所要求的删除操作。
4. 源代码清单：
5. #include<stdio.h>
6. void strcate(char [],char []);
7. void strdelc(char [],char );
8. int main(void) {
9. char a[]="Language", b[30]="Programming"; // 为连接的字符串留出足够的空间
10. printf("%s %s\n", b,a);
11. strcate(b,a);
12. printf("%s %s\n",b,a);
13. strdelc(b, 'a');
14. printf("%s\n",b);
15. return 0;
16. }
17. void strcate(char t[],char s[]) {
18. int i = 0, j = 0;
19. while(t[i++]) ;
20. i--; // 退回到'\0'的位置
21. while((t[i++] = s[j++] )!= '\0');
22. }
23. void strdelc(char s[], char c) {
24. int j,k;
25. for(j=k=0; s[j] != '\0'; j++) {
26. if(s[j] != c)
27. s[k++] = s[j];
28. }
29. s[k] = '\0'; // 为最后一位添加终止符
30. }
31. 错误修改后运行结果如图5-1所示。

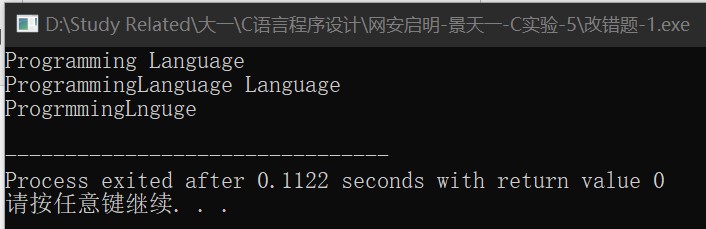


图5-1 实验5-1修改后运行结果截图

与预期结果相同，说明源代码正确。

**5.2.2 源程序完善和修改替换**

下面的源程序用于求解瑟夫问题：M个人围成一圈，从第一个人开始依次从1至N循环报数，每当报数为N时报数人出圈，直到圈中只剩下一个人为止。①请在源程序中的下划线处填写合适的代码来完善该程序。②上面的程序中使用数组元素的值表示圈中人的编号，故每当有人出圈时都要压缩数组，这种算法不够精炼。如果采用做标记的办法，即每当有人出圈时对相应数组元素做标记，从而可省掉压缩数组的时间，这样处理效率会更高一些。请采用做标记的办法修改程序，并使修改后的程序与原程序具有相同的功能。

1. #include<stdio.h>
2. #define M 10
3. #define N 3
4. int main(void)
5. {
6. int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/
7. int i, j, k;
8. for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/
9. a[i] = i + 1;
10. for(i = M, j = 0; i > 1; i--){
11. /\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/
12. for(k = 1; k <= N; k++) /\* 1至N报数 \*/
13. if(++j > i - 1) j = 0;/\* 最后一个人报数后第一个人接着报，形成一个圈 \*/
14. b[M-i] = j ? \_\_\_\_\_\_\_:\_\_\_\_\_\_; /\* 将报数为N的人的编号存入数组b \*/
15. if(j)
16. for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/
17. \_\_\_\_\_\_\_\_\_\_\_\_\_\_;
18. }
19. for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/
20. printf(“%6d”, b[i]);
21. printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/
22. return 0;
23. }

**解答：**

1. 在下划线处填写合适的语句：
2. 第一处：a[j-1]:a[i-1]
3. 第二处：a[k] = a[k+1]
4. 修改后的运行结果如图5-2所示：

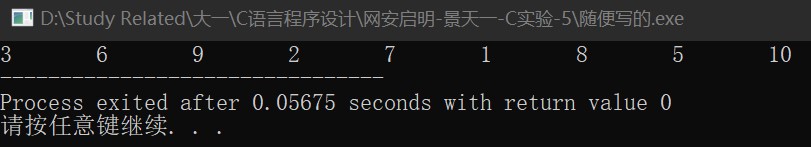


图5-2 实验5-2修改后运行结果截图

1. 按照要求优化程序：
2. 设计思路：

在一个数组中以0或1储存出圈或非出圈的状态，用数组中元素的位置表示编号；每次循环时，将出圈者的状态由1改为0进行标记；若循环位置状态为0，则跳过该位置。

1. 源程序清单：
2. #include<stdio.h>
3. #define M 10
4. #define N 3
5. int main(void) {
6. int lst[M]= {0}, return\_list[M-1]={}, count = 1, out\_num = 0;
7. for (int i=0; (i<M && out\_num<M); i++) {
8. if (!lst[i]) {
9. if (count==N) {
10. lst[i] = 1;
11. return\_list[out\_num] = i + 1;
12. out\_num++;
13. count = 1;
14. } else {
15. if (i == M-1)
16. i=-1;
17. count++;
18. }
19. }
20. }
21. for (int i=0; i<M-1; i++)
22. printf("%d\t",return\_list[i]);
23. return 0;
24. }
25. 优化和替换后运行结果如图5-3所示。

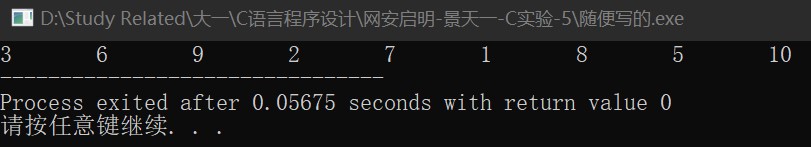


图5-3实验5-2优化替换后运行结果截图

**5.2.3 程序设计**

1. 输入一个整数，将它在内存中二进制表示的每一位转化成对应的数字字符并且存放到一个字符数组中，然后输出该整数的二进制表示。

**解答：**

1. 算法流程如图5-4所示。

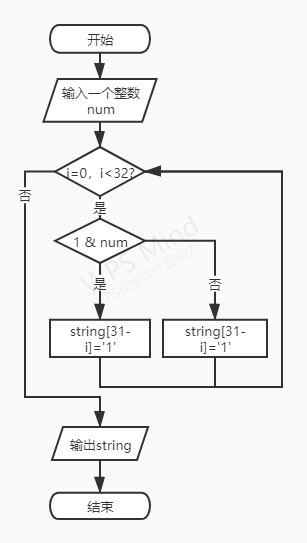


图5-4 程序设计题1的程序流程图

1. 源程序清单
2. #include <stdio.h>
3. int main(void)
4. {
5. int num;
6. char string[33];
7. int m, i;
8. printf("请输入要计算的数字：\n");
9. scanf("%d", &num);
10. for (i = 0, m = 1; i < 32; i++) {
11. string[31-i] = m & num ? '1' : '0'; //将结果存入一个字符数组中
12. m <<= 1; //对每一位用逻辑尺得到结果
13. }
14. string[32] = '\0';//将字符数组转变为字符串
15. printf("%s", string);//打印结果
16. return 0;
17. }
18. 测试

（a） 测试数据：

编程设计题1的测试数据如表5-1所示：

表5-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 1 | 00000000000000000000000000000001 |
| 用例2 | -1 | 11111111111111111111111111111111 |
| 用例3 | -2 | 11111111111111111111111111111110 |

（b） 对应测试测试用例1的运行结果如图5-5所示。



图5-5 程序设计题1的测试用例一的运行结果

对应测试测试用例2的运行结果如图5-6所示。

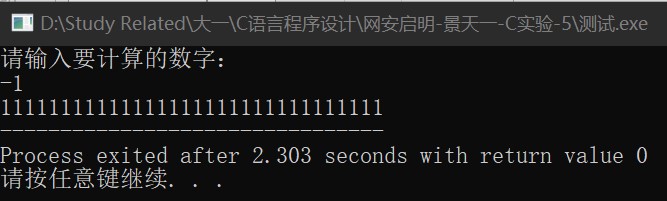


图5-6 程序设计题1的测试用例二的运行结果

对应测试测试用例3的运行结果如图5-7所示。

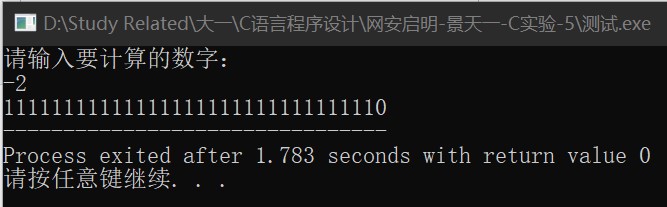


图5-7 程序设计题1的测试用例三的运行结果

1. 编写一个C程序，要求采用模块化程序设计思想，将相关功能用函数实现，并提供菜单选项。该程序具有以下功能：

①输入n个学生的姓名和C语言课程的成绩。

②将成绩按从高到低的次序排序，姓名同时进行相应调整。

③输出所有学生的姓名和C语言课程的成绩。

1. 对程序设计第（2）题的程序增加查找功能：输入一个C语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和C语言课程的成绩；否则，输出提示“not found!”。

**解答：**

1. 解题思路：
2. 分别输入学生成绩和姓名；
3. 按照成绩从高到低进行选择排序；
4. 输出指令菜单：输入指令以选择输出姓名成绩或按成绩查询姓名；
5. 若按成绩查询姓名：利用二分查找查询某一成绩对应的学生姓名（之一），再分别向前向后查找同成绩的学生，打印对应姓名，若成绩不相同则停止查找；
6. 程序清单
7. #include<stdio.h>
8. #include<stdlib.h>
9. #define SWAP(a,b) a=(a-b), b=(a+b), a=(b-a);
10. // 函数模块声明
11. int main(void); // 主函数
12. void menu(int [], char (\*)[10]); // 目录函数
13. void input(int [], char (\*)[10]); // 输入姓名，成绩
14. void output(int [], char (\*)[10]); // 输出成绩，姓名
15. void range(int [], char (\*)[10]); // 排序函数
16. void search(int [], char (\*)[10]); // 按照姓名搜索成绩的函数
17. // 实用函数声明
18. void repeatPrint(int num, char ch, char end); // 用于重复打印的函数
19. // 外部变量声明
20. int listlen=0; // 用于定义数组长度的变量
21. // 主函数
22. int main(void) {
23. repeatPrint(80,'=','\n');
24. printf("\n\t\t\t# 学生成绩排名与查询系统\n\n\t\t\t# 输入 1 进行成绩录入\n\t\t\t# 输入 其他字符 退出\n\n");
25. repeatPrint(80,'=','\n');
26. int order;
27. printf("$ >");
28. scanf("%d",&order);
29. if (order==1) {
30. printf("#请输入学生人数：");
31. scanf("%d",&listlen);
32. int scores[listlen];
33. char names[listlen][10];
34. input(scores, names);
35. range(scores, names);
36. while(1)
37. menu(scores, names);
38. } else
39. exit(1);
40. return 0;
41. }
42. // 目录函数
43. void menu(int scores[], char (\*names)[10]) {
44. repeatPrint(80,'=','\n');
45. printf("\n\t\t\t# 学生成绩排名与查询系统\n\n\t\t\t# 输入 2 进入成绩查询\n\t\t\t# 输入 3 输出成绩排名\n\t\t\t# 输入 其他字符 退出\n\n");
46. repeatPrint(80,'=','\n'); // #Step1：输入学生人数\n\t#Step2：输入“[学生姓名 学生成绩]
47. int order;
48. printf("$ >");
49. scanf("%d",&order);
50. system("cls");
51. if (order==2)
52. search(scores, names);
53. else if (order==3)
54. output(scores, names);
55. else
56. exit(1);
57. }
58. // 输入姓名成绩的函数
59. void input(int scores[], char (\*names)[10]) {
60. char ch=0;
61. for (int i=0; i<listlen; i++) {
62. printf("\n学生姓名：");
63. scanf("%s",names[i]);
64. printf("学生成绩：");
65. scanf("%d",&scores[i]);
66. }
67. }
68. // 排序函数(选择排序)
69. void range(int lst[], char (\*names)[10]) {
70. int n=listlen;
71. int count=0, count2=1;
72. for (count=0; count<n; count++) {
73. for (count2=count+1; count2<n; count2++) {
74. if (lst[count]>lst[count2]) {
75. SWAP(lst[count], lst[count2]);
76. for (int i=0; i<9; i++) {
77. SWAP(names[count][i], names[count2][i]);
78. }
79. }
80. }
81. }
82. }
83. // 用于输出的函数
84. void output(int scores[], char (\*names)[10]) {
85. for (int i=0; i < listlen; i++)
86. printf("姓名：%s\t成绩：%d\n",names[i], scores[i]);
87. }
88. // 用于重复打印的函数
89. void repeatPrint(int num, char ch, char end) {
90. for (int n=1; n<num; n++)
91. printf("%c",ch) ;
92. printf("%c",end);
93. }
94. // 用于搜索的函数
95. void search(int scores[], char (\*names)[10]) {
96. int high = listlen-1, low = 0, mid, num = 0, target;
97. printf("输入要查询的成绩：");
98. scanf("%d",&target);
99. for(mid = (low + high)/2;; mid = (low + high)/2) {
100. if (scores[high]==target) {
101. mid = high;
102. break;
103. } else if (scores[mid]==target) {
104. break;
105. } else if (low >= high || mid == high || mid == low) {
106. num = 1;
107. break;
108. } else if (scores[mid]>target) {
109. high = mid;
110. } else if(scores[mid]<target) {
111. low = mid;
112. }
113. }
114. if (num) {
115. printf("未找到相关信息！\n");
116. } else {
117. while ((mid>=low)&&(scores[mid-num]==target)) {
118. printf("姓名：%s\t成绩：%d\n",names[mid-num],target);
119. num++;
120. }
121. num=1;
122. while ((mid<=high)&&(scores[mid+num]==target)) {
123. printf("姓名：%s\t成绩：%d\n",names[mid+num],target);
124. num++;
125. }
126. }
127. }

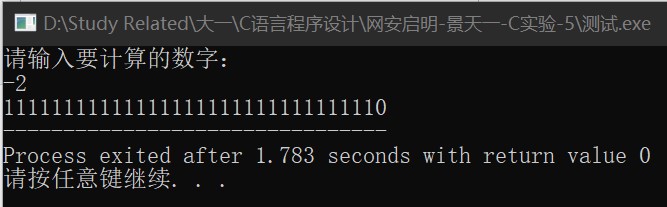
3）测试

（a） 测试数据如表5-2所示：

表5-2 编程题2，3的测试数据

|  |  |
| --- | --- |
| 测试用例 | 输入数据 |
| 用例1 | 啊啊啊 90  不不不 89  冲冲冲 94 |
| 用例2 | 啊啊啊 90  蛤蛤蛤 90  大大大 87  冲冲冲 96 |

（b） 对应测试测试用例1的运行结果如图5-8，5-9，5-10所示。



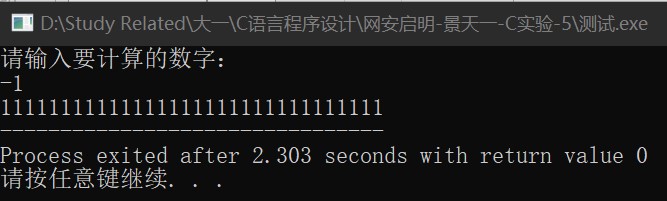


图5-8，5-9，5-10 程序设计题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图5-11，5-12，5-13所示。

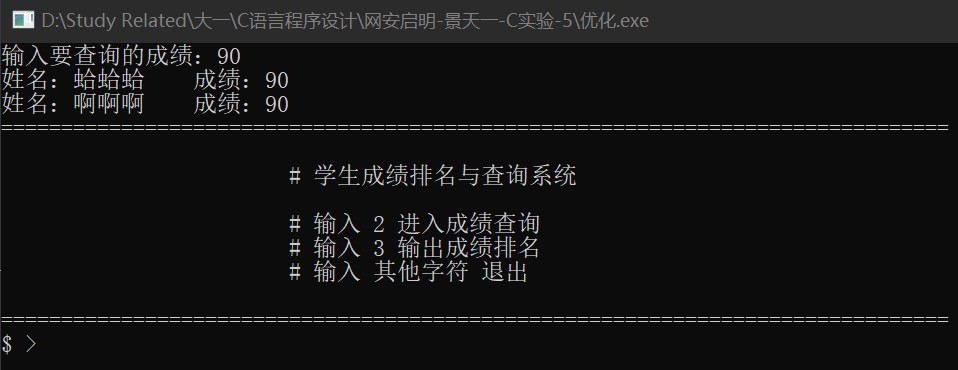
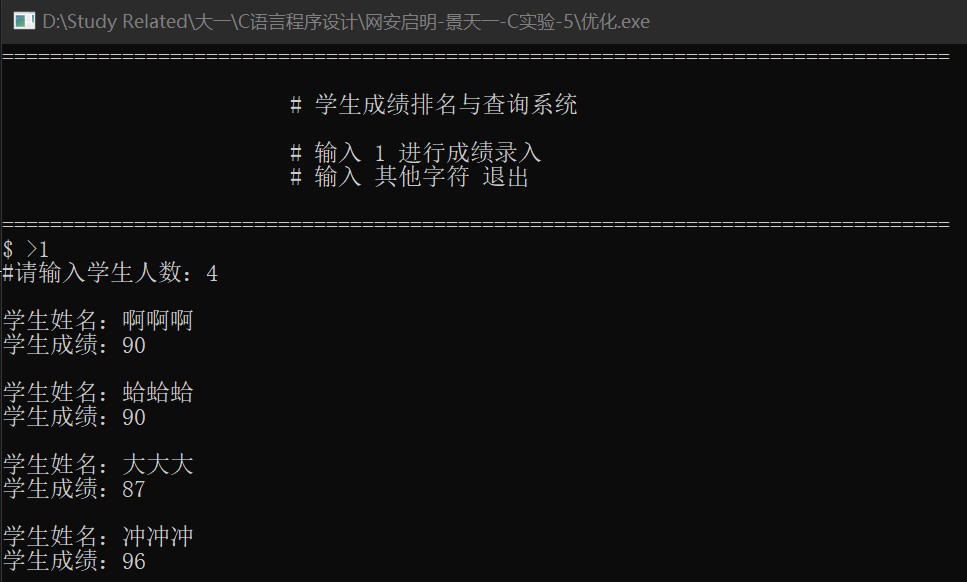


图5-11，5-12 程序设计题2的测试用例二的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**5.2.4 程序设计选做题**

编写并上机调试运行能实现以下功能的函数和程序。

编写函数strnins(s,t,n),其功能是：可将字符数组 t中的字符串插入到字符数组 s中字符串的第n个字符的后面。

**解答：**

1. 设计思路：
2. 在strins函数中新建一个长度为字符串s和t字符长度加1的数组用于返回数据；
3. 每次读取s字符数组的位置，当位置i小等于n时，依次输入s数组的字符；
4. 当位置i等于n小于n+ t 数组长度时，依次输入t数组字符；
5. 当位置i大于n + t数组长度时，继续输入s数组字符；
6. 为最后一位添加终止符’\0’；
7. 源代码清单：
8. #include<stdio.h>
9. #include<string.h>
10. void strnins(char [], char [], int );
11. int main(void) {
12. char s[]={'1','2','3','7','8','9'}, t[]={'4','5','6'};
13. int n=3;
14. strnins(s, t, n);
15. return 0;
16. }
17. void strnins(char s[], char t[], int n) {
18. char return\_list[strlen(s) + strlen(t) + 1];
19. for (int i = 0; i <= strlen(return\_list); i++) {
20. if (i < n) {
21. return\_list[i] = s[i];
22. } else if (i >= n && i < n + strlen(t)) {
23. return\_list[i] = t[i-n];
24. } else if (i >= n + strlen(t)) {
25. return\_list[i] = s[i-strlen(t)];
26. }
27. }
28. printf("%s\n", return\_list);
29. }
30. 测试

选做题1的运行结果如图5-13所示：



图5-13 选做题1的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 5.3 实验小结

本次实验主要练习了数组的使用，涉及数组的初始化，数组元素的赋值，作为函数参数的数组，字符数组的使用，作为函数参数的多维数组等。

其中，我对作为函数参数的多维数组的使用体会最深。在C语言中，传入函数的多维数组要求有确定的非第一唯长度，因此对多维数组的使用存在很多限制。在上网查询并询问老师后，了解到可以用一维指针或指向多维数组的指针的方式传入，引发了我对指针使用的好奇。同时，C语言对数组使用的诸多限制也要求我们寻找实现动态数组的方法。