# 6 指针实验

## 6.1 实验目的

（1）熟练掌握指针的说明、赋值、使用。

（2）掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

（3）熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

（4）掌握指针函数与函数指针的用法。

（5）掌握带有参数的main函数的用法。

## 6.2 实验题目及要求

**6.2.1 源程序改错题**

在下面所给的源程序中，函数strcopy(t, s)的功能是将字符串s复制给字符串t，并且返回串t的首地址。请单步跟踪程序，根据程序运行时出现的现象或观察到的字符串的值，分析并排除源程序的逻辑错误，使之能按照要求输出如下结果：

Input a string:

programming↙ （键盘输入）

programming

Input a string again:

language↙ （键盘输入）

language

1. #include<stdio.h>
2. char \*strcopy(char \*, const char \*);
3. int main(void)
4. {
5. char \*s1, \*s2, \*s3;
6. printf("Input a string:\n", s2);
7. scanf("%s", s2);
8. strcopy(s1, s2);
9. printf("%s\n", s1);
10. printf("Input a string again:\n", s2);
11. scanf("%s", s2);
12. s3 = strcopy(s1, s2);
13. printf("%s\n", s3);
14. return 0;
15. }
16. /\*将字符串s复制给字符串t，并且返回串t的首地址\*/
17. char \* strcopy(char \*t, const char \*s)
18. {
19. while(\*t++ = \*s++);
20. return (t);
21. }

**解答：**

1. 逻辑错误修改：
2. 第4行声明指针变量前，需要为指针提供具体指向的地址，防止出现野指针，正确形式为：

char ch\_1[100], ch\_2[100], ch\_3[100];

1. 第20行返回的指针t指向的是复制出的字符串的末地址，需要返回首地址，故添加一个字符指针变量指向首地址作为返回值：

char \*p=t;

while(\*p++ = \*s++);

return (t); // 返回字符串首地址

1. 修改后，源程序清单如下：
2. #include<stdio.h>
3. char \*strcopy(char \*, const char \*);
4. int main(void) {
5. char ch\_1[100], ch\_2[100], ch\_3[100]; // 为指针提供具体指向的变量，防止出现野指针情况
6. char \*s1=ch\_1, \*s2=ch\_1, \*s3=ch\_3;
7. printf("Input a string:\n", s2);
8. scanf("%s", s2);
9. strcopy(s1, s2);
10. printf("%s\n", s1);
11. printf("Input a string again:\n", s2);
12. scanf("%s", s2);
13. s3 = strcopy(s1, s2);
14. printf("%s\n", s3);
15. return 0;
16. }
17. /\*将字符串s复制给字符串t，并且返回串t的首地址\*/
18. char \* strcopy(char \*t, const char \*s) {
19. char \*p=t;
20. while(\*p++ = \*s++);
21. return (t); // 返回字符串首地址
22. }
23. 错误修改后运行结果如图6-1所示。

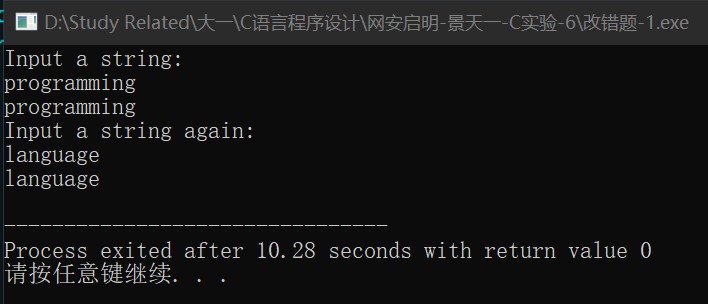


图6-1 实验6-1修改后运行结果截图

**6.2.2 源程序完善、修改替换题**

1. 下面程序中函数strsort用于对字符串进行升序排序，在主函数中输入N个字符串（字符串长度不超过49）存入通过malloc动态分配的存储空间，然后调用strsort对这N个串按字典序升序排序。

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。

1. #include<stdio.h>
2. #include<\_\_\_\_\_\_\_\_\_>
3. #include<string.h>
4. #define N 4
5. /\*对指针数组s指向的size个字符串进行升序排序\*/
6. void strsort(char \*s[], int size)
7. {
8. \_\_\_\_\_\_\_temp;
9. int i, j;
10. for(i=0; i<size-1; i++)
11. for (j=0; j<size-i-1; j++)
12. if (\_\_\_\_\_\_\_\_\_\_\_)
13. {
14. temp = s[j];
15. \_\_\_\_\_\_\_\_\_\_\_;
16. s[j+1] = temp;
17. }
18. }
19. int main()
20. {
21. int i;
22. char \*s[N], t[50];
23. for (i=0; i<N; i++)
24. {
25. gets(t);
26. s[i] = (char \*)malloc(strlen(t)+1);
27. strcpy(\_\_\_\_\_\_\_);
28. }
29. strsort(\_\_\_\_\_\_\_\_);
30. for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}
31. return 0;
32. }

②数组作为函数参数其本质类型是指针。例如，对于形参char \*s[]，编译器将其解释为char \*\*s，两种写法完全等价。请用二级指针形参重写strsort函数，并且在该函数体的任何位置都不允许使用下标引用。

**解答：**

1. 下划线处填空：
2. 第2行：#include<malloc.h>
3. 第7行：char \*temp;
4. 第11行：strcmp(s[j],s[j+1])==1
5. 第25行：strcpy(s[i], t);
6. 按要求修改后的源程序代码清单如下：
7. #include<stdio.h>
8. #include<malloc.h> //
9. #include<string.h>
10. #define N 4
11. /\*对指针数组s指向的size个字符串进行升序排序\*/
12. void strsort(char \*\*s, int size) {
13. char \*temp; //
14. int i, j;
15. for(i=0; i<size-1; i++) {
16. for (j=0; j<size-i-1; j++) {
17. if (strcmp(\*(s+j),\*(s+j+1))==1) { //
18. temp = \*(s+j);
19. \*(s+j) = \*(s+j+1);
20. \*(s+j+1) = temp;
21. }
22. }
23. }
24. }
25. int main() {
26. int i;
27. char \*s[N], t[50];
28. for (i=0; i<N; i++) {
29. gets(t);
30. s[i] = (char \*)malloc(strlen(t)+1);
31. strcpy(s[i], t); //
32. }
33. strsort(s, N);
34. for (i=0; i<N; i++) {
35. puts(s[i]);
36. free(s[i]);
37. }
38. return 0;
39. }
40. 测试用例如表6-1所示：

表6-1改错题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 测试用例1 | acdb  abcd  acbd  bcda | abcd  acbd  acdb  bcda |

填空后运行结果如图6-2所示。

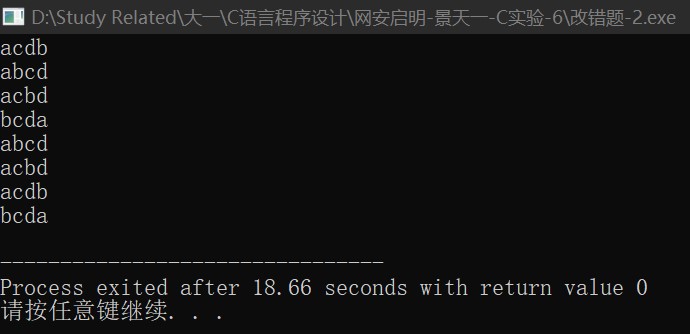


图6-2 实验6-2填空后运行结果截图

1. 下面源程序通过函数指针和菜单选择来调用库函数实现字符串操作；串复制strcpy、串连接strcat或串分解strtok。

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，使之能按照要求输出下面结果：

1 copy string.

2 connect string.

3 parse string.

4 exit.

input a number (1-4) please!

2↙ （键盘输入）

input the first string please!

the more you learn,↙ （键盘输入）

input the second string please!

the more you get. ↙ （键盘输入）

the result is the more you learn, the more you get.

1. # include<stdio.h>
2. # include<string.h>
3. int main (void)
4. {
5. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;
6. char a[80], b[80], \*result;
7. int choice;
8. while(1)
9. {
10. do
11. {
12. printf("\t\t1 copy string.\n");
13. printf("\t\t2 connect string.\n");
14. printf("\t\t3 parse string.\n");
15. printf("\t\t4 exit.\n");
16. printf("\t\tinput a number (1-4) please.\n");
17. scanf("%d", &choice);
18. }while(choice<1 || choice>4);
19. switch(choice)
20. {
21. case 1: p = strcpy; break;
22. case 2: p = strcat; break;
23. case 3: p = strok; break;
24. case 4: goto down;
25. }
26. getchar();
27. printf("input the first string please!\n");
28. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;
29. printf("input the second string please!\n");
30. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;
31. result = \_\_\_\_\_\_\_\_\_\_\_(a, b);
32. printf("the result is %s\n", result);
33. }
34. down:
35. return 0;
36. }

②函数指针的一个用途是用户散转程序，即通过一个转移表（函数指针数组）来实现多分枝函数处理，从而省去了大量的if语句或者switch语句。转移表中存放了各个函数的入口地址（函数名），根据条件的设定来查表选择执行相应的函数。请使用转移表而不是switch语句重写以上程序。

**解答：**

1. 下划线处填空：
2. 第4行：char \*(\*p)(char\*, const char \*);
3. 第31行：gets(a);
4. 第33行：gets(b);
5. 第34行：result = p(a, b);
6. 重写源程序后的代码清单如下：
7. # include<stdio.h>
8. # include<string.h>
9. int main (void) {
10. char \*(\*p)(char\*, const char \*); //
11. char a[80], b[80], \*result;
12. int choice;
13. while(1) {
14. do {
15. printf("\t\t1 copy string.\n");
16. printf("\t\t2 connect string.\n");
17. printf("\t\t3 parse string.\n");
18. printf("\t\t4 exit.\n");
19. printf("\t\tinput a number (1-4) please.\n");
20. scanf("%d", &choice);
21. } while(choice<1 || choice>4);
22. switch(choice) {
23. case 1:
24. p = strcpy;
25. break;
26. case 2:
27. p = strcat;
28. break;
29. case 3:
30. p = strtok;
31. break;
32. case 4:
33. goto down;
34. }
35. getchar();
36. printf("input the first string please!\n");
37. gets(a);
38. printf("input the second string please!\n");
39. gets(b);
40. result = p(a, b);
41. printf("the result is %s\n", result);
42. }
43. down:
44. return 0;
45. }
46. 用函数指针作为转移表改写程序后的源代码如下：
47. # include<stdio.h>
48. # include<string.h>
49. int main (void) {
50. char \*(\*p[3])(char\*, const char \*)={
51. strcpy,
52. strcat,
53. strtok,
54. }; //
55. char a[80], b[80], \*result;
56. int choice;
57. while(1) {
58. do {
59. printf("\t\t1 copy string.\n");
60. printf("\t\t2 connect string.\n");
61. printf("\t\t3 parse string.\n");
62. printf("\t\t4 exit.\n");
63. printf("\t\tinput a number (1-4) please.\n");
64. scanf("%d", &choice);
65. } while(choice<1 || choice>4);
66. if (choice == 4)
67. goto down;
68. getchar();
69. printf("input the first string please!\n");
70. gets(a);
71. printf("input the second string please!\n");
72. gets(b);
73. result = p[choice-1](a, b);
74. printf("the result is %s\n", result);
75. }
76. down:
77. return 0;
78. }
79. 测试：

源程序按照题目要求运行后的结果如图6-3所示：

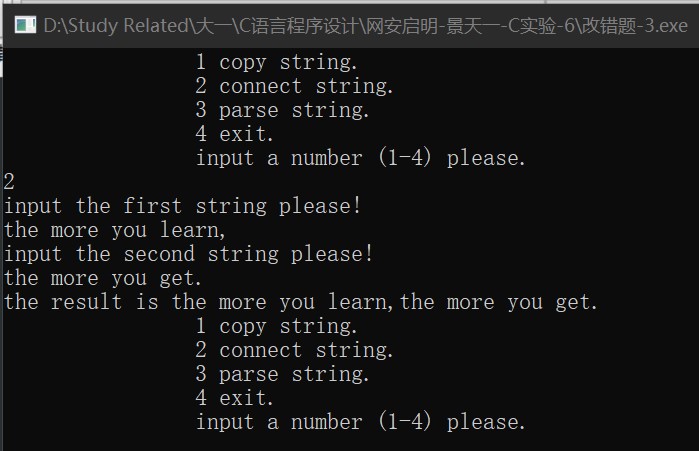


图6-3 实验6-3填空并修改后运行结果截图

**6.2.3 跟踪调试题**

请按下面的要求对源程序进行操作，并回答问题和排除错误。

（1）单步执行。进入strcpy时watch窗口中s为何值？返回main时, watch窗口中s为何值？

（2）排除错误，使程序输出结果为：there is a boat on the lake.

1. #include "stdio.h"
2. char \*strcpy(char \*,char \*);
3. void main(void)
4. {
5. char a[20],b[60]="there is a boat on the lake.";
6. printf("%s\n",strcpy(a,b));
7. }
8. char \*strcpy(char \*s,char \*t)
9. {
10. while(\*s++=\*t++)
11. ;
12. return (s);
13. }

**解答：**

1. 单步执行： 进入strcpy时watch窗口中s为ox62fe00””；返回main时, watch窗口中s为0x62fe1d””；
2. 程序改错：第9行添加char \*p=s; 第10行正确形式为：while(\*p++=\*t++); 最后返回的即为s的首地址；
3. 源程序清单：
4. #include "stdio.h"
5. char \*strcpy(char \*,char \*);
6. int main(void) {
7. char a[20],b[60]="there is a boat on the lake.";
8. printf("%s\n",strcpy(a,b));
9. return 0;
10. }
11. char \*strcpy(char \*s,char \*t) {
12. char \*p=s;
13. while(\*p++=\*t++);
14. return (s);
15. }
16. 测试：

跟踪调试题1的运行结果如图6-4所示：

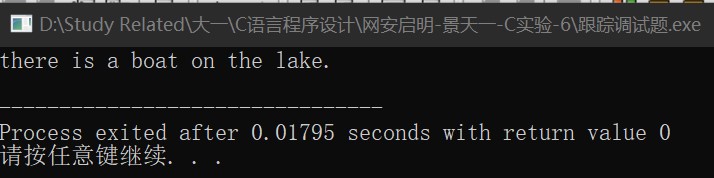


图6-4 跟踪调试题修改后运行结果截图

**6.2.4 程序设计**

1. 一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。试从该长整型变量的高字节开始，依次取出每个字节的高4位和低4位并以十六进制数字字符的形式进行显示，要求通过指针取出每字节。

**解答：**

1. 解题思路：

程序设计题-1的解题思路如图6-5所示：

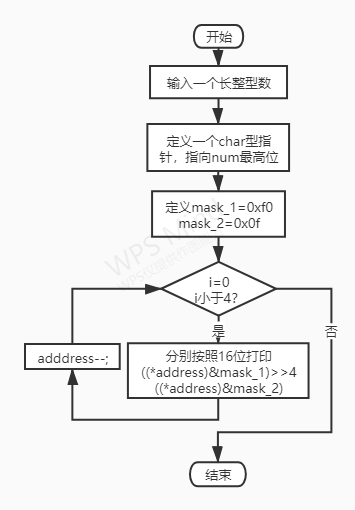


图6-5 程序设计题1的程序流程图

1. 源程序清单
2. #include<stdio.h>
3. int main(void) {
4. long int num=0x1234ABCD;
5. scanf("%ld", &num);
6. char \*address = (char\*)&num;
7. address += 3;
8. int mask\_1=0xf0, mask\_2=0x0f;
9. for (int i = 0; i < 4; i++) {
10. printf("0x%x\t", ((\*address)&mask\_1)>>4);
11. printf("0x%x\n", ((\*address)&mask\_2));
12. address--;
13. }
14. return 0;
15. }
16. 运行结果：

设计题-1的运行结果如图6-6所示：

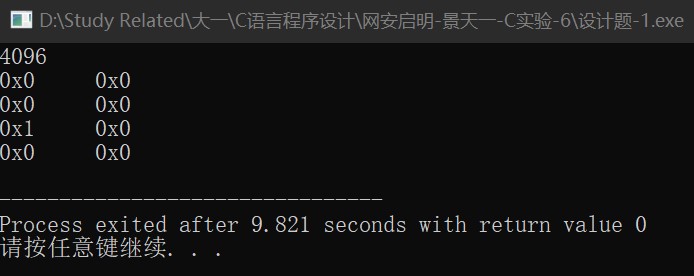


图6-6 设计题1运行结果截图

1. 旋转是图像处理的基本操作，编程实现一个将一个图像逆时针旋转90°。提示：计算机中的图像可以用一个矩阵来表示，旋转一个图像就是旋转对应的矩阵。将旋转矩阵的功能定义成函数，通过使用指向数组元素的指针作为参数使该函数能处理任意大小的矩阵。要求在main函数中输入图像矩阵的行数n和列数m，接下来的n行每行输入m个整数，表示输入的图像。输出原始矩阵逆时针旋转90°后的矩阵。例如，输入：

2 3

1 5 3

3 2 4

则输出：

3 4

5 2

1 3

**解答：**

1. 解题思路：

设计题-2的解题思路如图6-7所示：

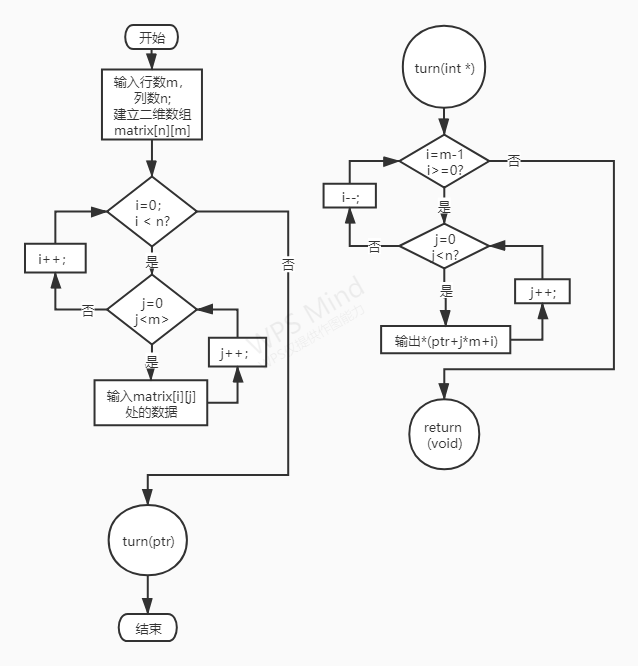


图6-7 程序设计题2的程序流程图

1. 源程序清单：
2. #include<stdio.h>
3. void turn(int \*);
4. int m, n;
5. int main(void) {
6. scanf("%d %d", &n, &m);
7. int matrix[n][m];
8. for (int i = 0; i < n; i++) {
9. for (int j = 0; j < m; j++) {
10. scanf("%d",&matrix[i][j]);
11. }
12. }
13. printf("\nInput complete!\n");
14. int \*ptr=(int\*)&matrix[0];
15. turn(ptr);
16. return 0;
17. }
18. void turn(int \*ptr) {
19. for (int i = m-1; i >= 0; i--) {
20. for (int j = 0; j < n; j++) {
21. // 希望打印matrix[j][i]
22. printf("%d ", \*(ptr + j\*m + i));
23. }
24. printf("\n");
25. }
26. }

3）测试

（a） 测试数据：

测试数据如表6-2所示。

表6-2 编程题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入数据 | 预计输出 |
| 用例1 | 2 3  1 5 3  3 2 4 | 3 4  5 2  1 3 |
| 用例2 | 5 4  3 0 9 7  8 9 6 3  0 4 2 7  2 9 5 6  9 6 5 3 | 7 3 7 6 3  9 6 2 5 5  0 9 4 9 6  3 8 0 2 9 |
| 用例3 | 1 4  11 76 88 45 | 45  88  76  11 |

（b） 对应测试测试用例1的运行结果如图6-8所示。

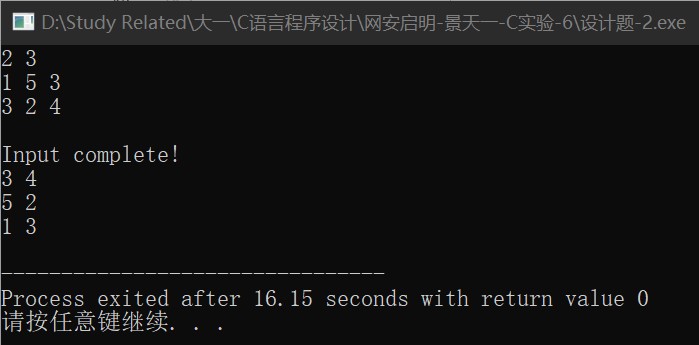


图6-8 程序设计题2的测试用例一的运行结果

对应测试测试用例2的运行结果如图6-9所示。



图6-9 程序设计题2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6-10所示。

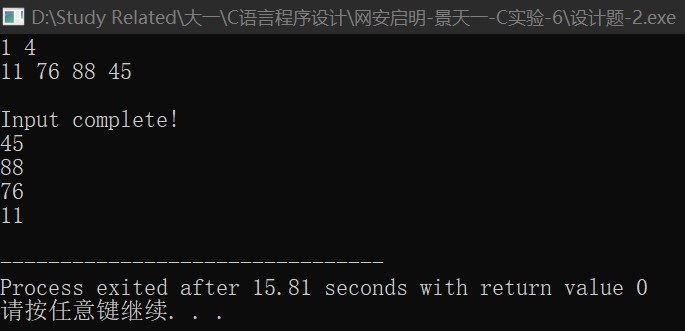


图6-10 程序设计题2的测试用例三的运行结果

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。

1. 输入n行文本，每行不超过80个字符，用字符指针数组指向键盘输入的n行文本，且n行文本的存储无冗余，删除每一行中的前置空格（' '）和水平制表符（'\t'）。要求：将删除一行文本中前置空格和水平制表符的功能定义成函数，在main函数中输出删除前置空格符的各行。

**解答：**

1. 程序设计题3的解题思路如图6-11所示：

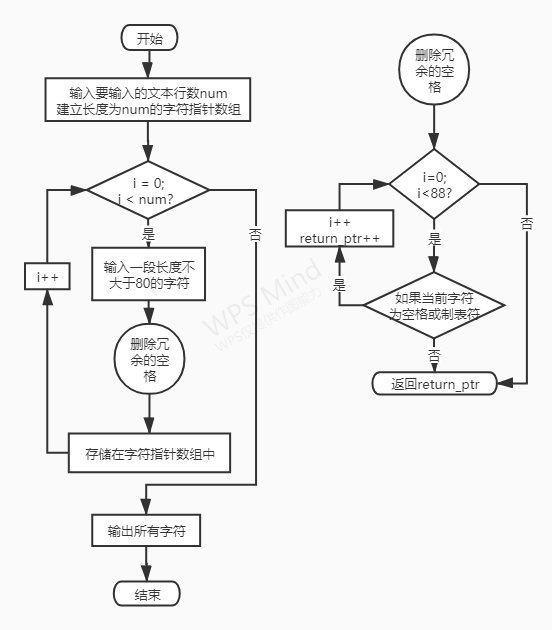


图6-11 程序设计题3的程序流程图

1. 源程序清单：
2. #include<stdio.h>
3. #include<malloc.h>
4. #include<string.h>
5. char\* delspc(char\*);
6. int main(void) {
7. int num=0;
8. scanf("%d", &num);
9. getchar();
10. char \*ch\_list[num], ch[80]={ch[79]='\0'}, \*ptr=ch;
11. for (int i=0; i<num; i++) {
12. gets(ch);
13. ptr = delspc(ptr);
14. ch\_list[i] = (char\*)malloc(strlen(ch)+1);
15. strcpy(ch\_list[i], ptr);
16. }
17. for (int i=0; i<num; i++) {
18. printf("%s\n", ch\_list[i]);
19. free(ch\_list[i]);
20. }
21. return 0;
22. }
23. char\* delspc(char \*ch) {
24. char \*return\_ptr=ch;
25. for (int i=0; i<80; i++) {
26. if (\*(ch+i)==' '||\*(ch+i)==' ') {
27. return\_ptr++;
28. } else {
29. break;
30. }
31. }
32. return return\_ptr;
33. }
34. 测试
35. 测试数据：

程序设计题3的测试数据如表6-3所示:

表6-3 编程题3的测试数据

|  |  |  |
| --- | --- | --- |
| 用例 | 实际输入 | 预计输出 |
| 用例1 | 3(回车)  A a(回车)  Aaa aa(回车)  A a(回车) | A a  Aaa aa  A a |

1. 对应测试测试用例1的运行结果如图6-12所示。



图6-12 程序设计题3的测试用例一的运行结果

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。

1. 编写8个任务函数，一个scheduler调度函数和一个execute执行函数。仅在main函数中调用scheduler函数，scheduler函数要求用最快的方式调度执行用户指定的任务函数。

①先设计task0, task1, task2, task3, task4, task5, task6, task7共8个任务函数，每个任务函数的任务就是输出该任务被调用的字符串。例如，第0个任务函数输出“task0 is called!”，第1个任务函数输出“task1 is called!”，以此类推。

②scheduler函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350并回车，则scheduler函数一次调度选择task1, task3, taks5, task0，然后以函数指针数组和任务个数为参数将调度选择结果传递给execute函数并调用execute函数。

③execute函数根据scheduler函数传递的指针数组和任务个数为参数，按照指定的先后顺序依此调用执行选定的任务函数。

例如，当输入13607122并回车，程序运行结果如下：

task1 is called!

task3 is called!

task6 is called!

task0 is called!

task7 is called!

task1 is called!

task2 is called!

task2 is called!

**解答:**

1. 程序设计题4的解题思路如下所示:
2. 声明一个有8个元素的函数指针数组,依次初始化为8个任务函数;
3. 以字符串形式输入指令;
4. 声明一个函数指针数组,长度为输入的指令的长度;
5. 对输入的指令中的每个字符,依次执行1中声明的函数指针数组的元素(任务函数);
6. 结束;
7. 源程序清单:
8. #include<stdio.h>
9. #include<string.h>
10. void task0(void) {
11. printf("task0 is called!\n");
12. }
13. void task1(void) {
14. printf("task1 is called!\n");
15. }
16. void task2(void) {
17. printf("task2 is called!\n");
18. }
19. void task3(void) {
20. printf("task3 is called!\n");
21. }
22. void task4(void) {
23. printf("task4 is called!\n");
24. }
25. void task5(void) {
26. printf("task5 is called!\n");
27. }
28. void task6(void) {
29. printf("task6 is called!\n");
30. }
31. void task7(void) {
32. printf("task7 is called!\n");
33. }
34. void execute(int num, void (\*fun[])(void)) {
35. for (int i = 0; i < num; i++) {
36. fun[i]();
37. }
38. }
39. void scheduler(void) {
40. void (\*func\_list[8])(void)={
41. task0,task1,task2,task3,task4,task5,task6,task7
42. };
43. char names[9];
44. scanf("%s", &names);
45. int num = strlen(names);
46. void (\*function[num])(void);
47. for (int i = 0; i < num; i++) {
48. function[i] = func\_list[(int)names[i]-48];
49. }
50. execute(num, function);
51. }
52. int main(void) {
53. scheduler();
54. return 0;
55. }
56. 测试:
57. 测试数据:

程序设计题4的测试数据如表6-4所示:

表6-4 编程题4的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 实际输入 | 预计输出 |
| 用例1 | 13607122 | task1 is called!  task3 is called!  task6 is called!  task0 is called!  task7 is called!  task1 is called!  task2 is called!  task2 is called! |

1. 对应测试测试用例1的运行结果如图6-13所示。

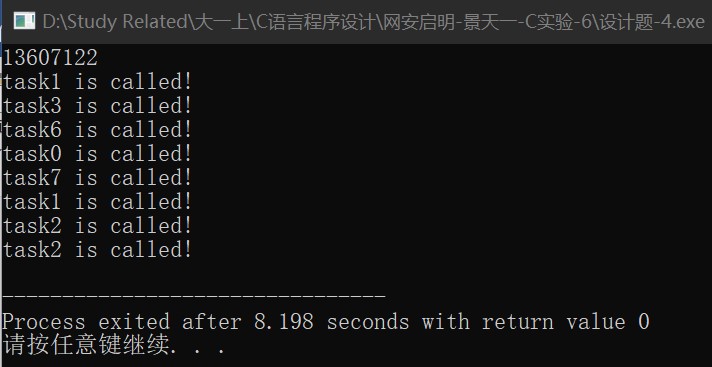


图6-13 程序设计题3的测试用例一的运行结果

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。

**6.2.5 选做题**

1. 设有N位整数和M位小数（N=20，M=10）的数据a,b。编程计算a+b并输出结果。

如：12345678912345678912.1234567891 + 765432109876543210.0123456789

**解答:**

1. 解题思路:

选做题1的解题思路如下所示:

1. 输入需要相加的两个数字分别存储在字符串num\_1,num\_2中;
2. 声明整型变量add=0表示进位;
3. 从低到高遍历两个数字的每一位,取两数与进位add相加后除以十的余数记为该位的字符,若相加大于十,则记进位add为1;
4. 为输出的字符串的最后一位添加’\0’;
5. 源代码清单:
6. #include<stdio.h>
7. #define N 20
8. #define M 10
9. int main(void) {
10. char num\_1[M+N+2], num\_2[M+N+2], result[M+N+3];
11. printf("Input the first number: ");
12. gets(num\_1);
13. printf("Input the second number:");
14. gets(num\_2);
15. result[N+1] = '.';
16. int add = 0;
17. for (int i = M+N; i >= 0; i--) {
18. if (num\_1[i]!='.') {
19. int num\_one = num\_1[i] - '0', num\_two = num\_2[i] - '0';
20. result[i+1] = (num\_one + num\_two + add)%10 + '0';
21. if ((num\_one + num\_two+ add)>=10)
22. add = 1;
23. else
24. add = 0;
25. }
26. }
27. result[0] = add + '0';
28. if (result[0]=='0')
29. printf("%s", result+1);
30. else
31. printf("%s", result);
32. return 0;
33. }
34. 测试

选做题1的运行结果如图6-14所示:

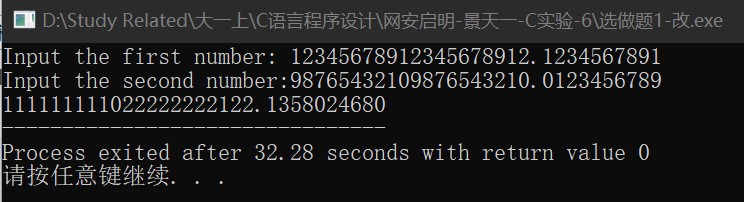


图6-14 选做题1的运行结果

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。

1. 编写使用复杂声明char \*(\*p[2])(const char \*,const char \*);的程序。提示：p中元素可为strstr等函数名。

**解答:**

1. 解题思路:

题目中声明了一个有两个元素的函数指针数组,指向参数为两个const char型指针,返回值为char型指针的函数;

1. 源程序清单:
2. #include<stdio.h>
3. #include<string.h>
4. char \*func(const char \*ch\_1,const char \*ch\_2){
5. printf("%s\n%s\n", ch\_1, ch\_2);
6. char return\_p[30];
7. strcpy(return\_p, ch\_1);
8. return return\_p;
9. }
10. int main(void) {
11. char str1[20],str2[20]={"Hello World!"},str3[20]={" World!"};
12. char \*(\*p[2])(const char \*,const char \*)={
13. strstr,
14. func,
15. };
16. printf("%s", p[1](str2, str3));
17. return 0;
18. }
19. 测试：
20. 选做题2的运行结果如图6-15所示:

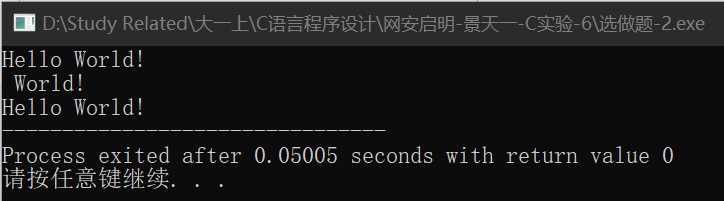


图6-15 选做题2的运行结果

1. 指定main函数的参数

在IDE（比如DevC++）中，选择“**运行**”｜“**参数**”菜单，在 “**传递给主程序的参数**”文本框中输入main函数的参数arg1 arg2 arg3，只输入命令行中文件名后的参数，文件名不作为参数输入，参数间以空格隔开。编写程序在命令行输出这三个参数。（注意不同IDE输入参数的方式不相同，可参考各个IDE的使用手册。）

**解答：**

1. 解题思路：

在main函数中打印输入的非文件名参数；

1. 源程序清单：
2. #include<stdio.h>
3. int main(int argc,char \*\*argv) {
4. for (int i=1; i<argc; i++) {
5. printf("%s\n",argv[i]);
6. }
7. return 0;
8. }
9. 测试：

向命令行输入参数的截图如图6-16所示：



图6-16 选做题3向命令行输入参数

选做题3的运行结果如图6-17所示：

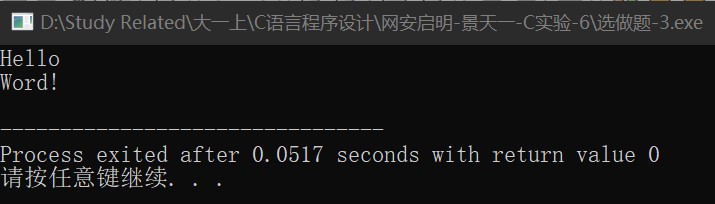


图6-17 选做题3的运行结果

## 6.3 实验小结

指针是C语言学习和使用的重难点。本次实验主要练习了指针的使用，包括指针变量的声明和引用，利用指针引用所指变量，作为函数参数和返回值的指针的使用，复杂类型指针的使用等。

指针常常被称为C语言的精华，这显示出了它的重要意义。借由指针，C语言可以对计算机内存进行直接操作，更加贴近底层，大大提高了执行效率。但同时，这也要求设计程序时格外重视对指针的使用，防止出现野指针，内存泄漏等情况。