

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期：**

**软件学院**

**目 录**

[6 指针实验 1](#_Toc183196209)

[6.1实验目的 1](#_Toc183196210)

[6.2实验题目及要求 1](#_Toc183196211)

[6.3实验小结 7](#_Toc183196215)

实验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

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. 错误修改
   1. 第5行中的s1 s2 s3均为野指针，应改为定长数组，即改为：  
      char s1[1000], s2[1000], s3[1000];
   2. 第12行中strcopy用法有误，可直接将s2复制到s3，即改为：  
      strcopy(s3, s2);
2. 修改后完整代码

#include<stdio.h>

char \*strcopy(char \*, const char \*);

int main(void)

{

char s1[1000], s2[1000], s3[1000];

printf("Input a string:\n", s2);

scanf("%s", s2);

strcopy(s1, s2);

printf("%s\n", s1);

printf("Input a string again:\n", s2);

scanf("%s", s2);

strcopy(s3, s2);

printf("%s\n", s3);

return 0;

}

/\*将字符串s复制给字符串t，并且返回串t的首地址\*/

char \* strcopy(char \*t, const char \*s)

{

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

return (t);

}

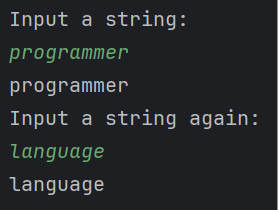
1. 修改后效果

图6-1 程序改错题的运行结果图

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（提示：使用strcmp函数比较字符（串）大小）。

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。

#include<stdio.h>

#include<\_\_\_\_\_\_\_\_\_>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*s[], int size)

{

\_\_\_\_\_\_\_temp;

int i, j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (\_\_\_\_\_\_\_\_\_\_\_)

{

temp = s[j];

\_\_\_\_\_\_\_\_\_\_\_;

s[j+1] = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

for (i=0; i<N; i++)

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(\_\_\_\_\_\_\_);

}

strsort(\_\_\_\_\_\_\_\_);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

**解答：**

1. 修改后完整代码

#include<stdio.h>  
#include<stdlib.h>  
#include<string.h>  
#define N 4  
/\*对指针数组s指向的size个字符串进行升序排序\*/  
void strsort(char \*s[], int size)  
{  
 char \*temp;  
 int i, j;  
 for(i=0; i<size-1; i++)  
 for (j=0; j<size-i-1; j++)  
 if (strcmp(s[j], s[j+1]) > 0)  
 {  
 temp = s[j];  
 s[j] = s[j+1];  
 s[j+1] = temp;  
 }  
}  
int main()  
{  
 int i;  
 char \*s[N], t[50];  
 for (i=0; i<N; i++)  
 {  
 gets(t);  
 s[i] = (char \*)malloc(strlen(t)+1);  
 strcpy(s[i], t);  
 }  
 strsort(s, N);  
 for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}  
 return 0;  
}

1. 图片包含 文本

   描述已自动生成修改后运行效果

图6-2 程序修改题1的运行结果图

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：**

void strsort(char \*\*s, int size)

{

char \*temp;

int i, j;

for(i=0; i<size-1; i++)

for (j=0; j<size-i-1; j++)

if (strcmp(\*(s+j), \*(s+j+1)) > 0)

{

temp = \*(s+j);

\*(s+j) = \*(s+j+1);

\*(s+j+1) = temp;

}

}

（2）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.

# include<stdio.h>

# include<string.h>

int main (void)

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

switch(choice)

{

case 1: p = strcpy; break;

case 2: p = strcat; break;

case 3: p = strtok; break;

case 4: goto down;

}

getchar();

printf("input the first string please!\n");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

printf("input the second string please!\n");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

result = \_\_\_\_\_\_\_\_\_\_\_(a, b);

printf("the result is %s\n", result);

}

down:

return 0;

}

**解答：**

1. 修改后完整代码

#include<stdio.h>  
#include<string.h>  
int main (void)  
{  
 char \*(\*p)(char \*, const char \*);  
 char a[80], b[80], \*result;  
 int choice;  
 while(1)  
 {  
 do  
 {  
 printf("\t\t1 copy string.\n");  
 printf("\t\t2 connect string.\n");  
 printf("\t\t3 parse string.\n");  
 printf("\t\t4 exit.\n");  
 printf("\t\tinput a number (1-4) please.\n");  
 scanf("%d", &choice);  
 }while(choice<1 || choice>4);  
 switch(choice)  
 {  
 case 1: p = strcpy; break;  
 case 2: p = strcat; break;  
 case 3: p = strtok; break;  
 case 4: goto down;  
 }  
 getchar();

printf("input the first string please!\n");

gets(a);

printf("input the second string please!\n");

gets(b);

result = p(a, b);

printf("the result is %s\n", result);

}  
 down:  
 return 0;  
}

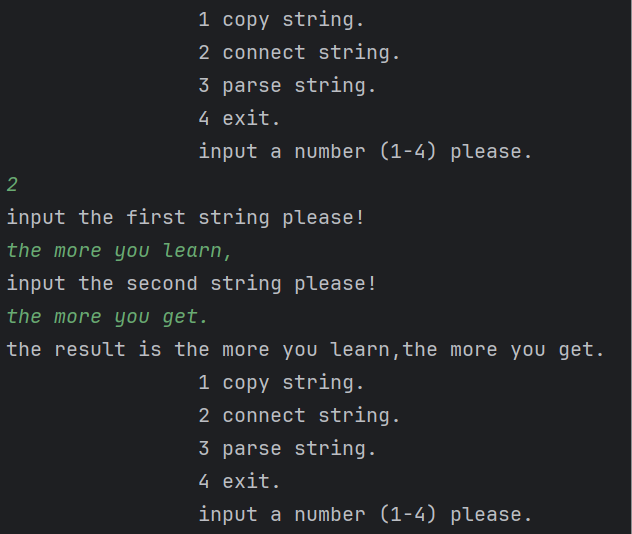
1. 修改后运行效果

图6-3 程序修改题2的运行结果图

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：**

#include<stdio.h>

#include<string.h>

int main (void)

{

char \*(\*p[])(char \*, const char \*) = {strcpy, strcat, strtok};

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

if (choice == 4) goto down;

getchar();

printf("input the first string please!\n");

gets(a);

printf("input the second string please!\n");

gets(b);

result = p[choice - 1](a, b);

printf("the result is %s\n", result);

}

down:

return 0;

}

3、编程设计题

（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. 源程序清单

#include<stdio.h>  
int main(int argc, char \*argv[])  
{  
 for (int i = 1; i < argc; i++) {  
 printf("%s\n", argv[i]);  
 }  
 return 0;  
}

1. 图片包含 文本

   描述已自动生成运行效果

图6-4 编程题1的运行结果图

以下（2）至（5）题对应Educoder 教学平台“C语言实验”课程，实验6，第13关实验6-1、第14关实验6-2、第15关实验6-3, 以及第16关实验6-4。

（2）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。输入一个长整型变量，要求从高字节开始，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，通过指针取出每字节。

样例输入：15

样例输出：0000000F

**解答：**

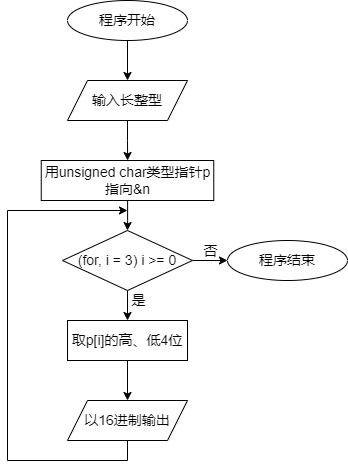
1. 算法流程

图6-5 编程题2的程序流程图

1. 源程序清单

#include<stdio.h>

int main() {

    long n;

    scanf("%ld", &n);

    unsigned char\* a = &n;

    for (int i = 3; i >= 0; i--) {

        short high = a[i] >> 4;

        short low = a[i] & 0xF;

        printf("%01X%01X", high, low);

    }

    return 0;

}

1. 测试数据

表6-1 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 15 | 0000000F | 0000000F |
| 用例2 | 2147483647 | 7FFFFFFF | 7FFFFFFF |

文本

描述已自动生成

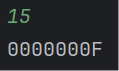
1. 运行效果

图6-6 编程题1的运行结果图

（3）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。

样例输入：

2 3

1 5 3

3 2 4

样例输出：

3 4

5 2

1 3

**解答：**

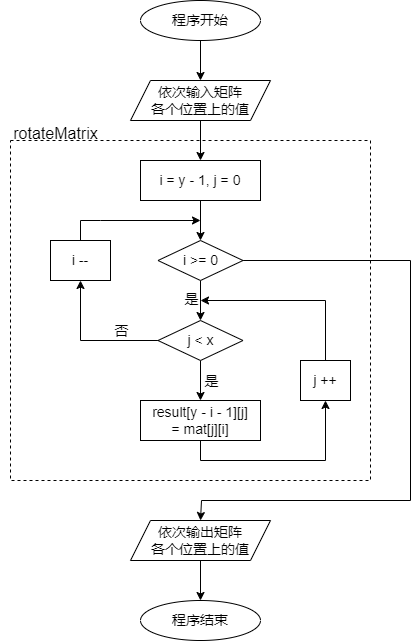
1. 算法流程

图6-7 编程题3的程序流程图

1. 源程序清单

#include<stdio.h>

void (int x, int y, int mat[x][y], int result[y][x]) {

    for (int i = y - 1; i >= 0; i--) {

        for (int j = 0; j < x; j++) {

            result[y - i - 1][j] = mat[j][i];

        }

    }

}

int main() {

    int x, y;

    scanf("%d %d", &x, &y);

    int mat[x][y];

    for (int i = 0; i < x; i++) {

        for (int j = 0; j < y; j++) {

            scanf("%d", &mat[i][j]);

        }

    }

    int result[y][x];

    rotateMatrix(x, y, mat, result);

    for (int i = 0; i < y; i++) {

        for (int j = 0; j < x; j++) {

            printf("%d", result[i][j]);

            if (j != x - 1) {

                printf(" ");

            }

        }

        printf("\n");

    }

    return 0;

}

1. 测试数据

表6-2 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 2 3  1 5 3  3 2 4 | 3 4  5 2  1 3 | 3 4  5 2  1 3 |
| 用例2 | 3 4  1 2 3 4  5 6 7 8  9 10 11 12 | 4 8 12  3 7 11  2 6 10  1 5 9 | 4 8 12  3 7 11  2 6 10  1 5 9 |

1. 图片包含 文本

   描述已自动生成文本

   中度可信度描述已自动生成运行效果

图6-8 编程题3的运行结果图

（4）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：**

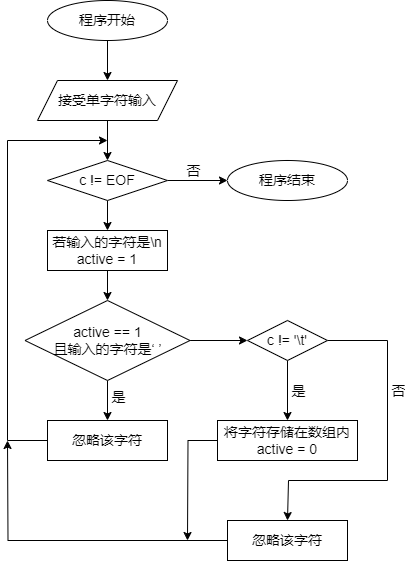
1. 算法流程

图6-9 编程题4的程序流程图

1. 源程序清单

#include<stdio.h>

char in[10000];

int active = 1;

int checkValidity(char c) {

    if (active && c != ' ') {

        active = 0;

    }

    if (c == '\n') {

        active = 1;

    }

    if (c == '\t') {

        return 0;

    }

    if (active && c == ' ') {

        return 0;

    }

    return 1;

}

int main() {

    int i = 0;

    while ((in[i] = getchar()) != EOF) {

        if (!checkValidity(in[i])) {

            continue;

        }

        i++;

    }

    in[i] = '\0';

    printf("%s", in);

    return 0;

}

1. 测试数据

表6-3 编程题4的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | aab c  bcd(\t)e | aab c  bcde | aab c  bcde |
| 用例2 | fasf(\t\t) 2rf  aa22(\t)e | fasf 2rf  aa22e | fasf 2rf  aa22e |

1. 手机屏幕的截图

   描述已自动生成手机屏幕的截图

   描述已自动生成运行效果

图6-10 编程题4的运行结果图

（5）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：**

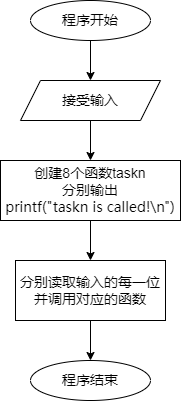
1. 算法流程

图6-11 编程题5的程序流程图

1. 源程序清单

#include<stdio.h>

void task0() {

    printf("task0 is called!\n");

}

void task1() {

    printf("task1 is called!\n");

}

void task2() {

    printf("task2 is called!\n");

}

void task3() {

    printf("task3 is called!\n");

}

void task4() {

    printf("task4 is called!\n");

}

void task5() {

    printf("task5 is called!\n");

}

void task6() {

    printf("task6 is called!\n");

}

void task7() {

    printf("task7 is called!\n");

}

void(\* tasks[])() = {&task0, &task1, &task2, &task3, &task4, &task5, &task6, &task7};

void execute(int count, void(\* waitlist[])()) {

    for (int i = 0; i < count; i++) {

        waitlist[i]();

    }

}

void scheduler() {

    void(\* waitlist[100])() = {};

    char s[100];

    int count = 0, index = 0;

    scanf("%s", s);

    while (s[index] != '\0') {

        waitlist[index] = tasks[s[index] - '0'];

        index++;

        count++;

    }

    execute(count, waitlist);

}

int main() {

    scheduler();

    return 0;

}

1. 测试数据

表6-4 编程题5的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 1235150177 | task1 is called!  task2 is called!  task3 is called!  task5 is called!  task1 is called!  task5 is called!  task0 is called!  task1 is called!  task7 is called!  task7 is called! | task1 is called!  task2 is called!  task3 is called!  task5 is called!  task1 is called!  task5 is called!  task0 is called!  task1 is called!  task7 is called!  task7 is called! |
| 用例2 | 72565132 | task7 is called!  task2 is called!  task5 is called!  task6 is called!  task5 is called!  task1 is called!  task3 is called!  task2 is called! | task7 is called!  task2 is called!  task5 is called!  task6 is called!  task5 is called!  task1 is called!  task3 is called!  task2 is called! |

1. 文本

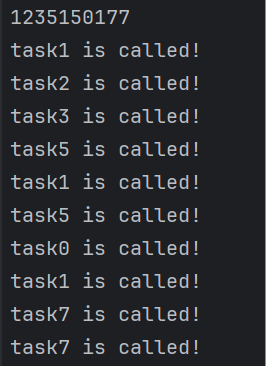
   描述已自动生成运行效果

图6-12 编程题5的运行结果图

6.3 实验小结

1. 在本次实验中，我学会了使用编译预处理命令，学到了宏定义、条件编译的使用。
2. 在本次实验中，我使用了debug功能对程序进行调试，使问题定位更加快速便捷。