

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 网安2003**

**学 号： U202012043**

**姓 名： 范启航**

**指导教师： 张云鹤**

**报告日期： 2020年12月5日**

**网络空间安全学院**

**目□□录**

[**1□□□表达式和标准输入输出实验 1**](#_Toc404837920)

[1.1□□实验目的 1](#_Toc404837921)

[1.2□□实验内容 1](#_Toc404837922)

[1.3□□实验小结 1](#_Toc404837923)

[**2□□□流程控制实验 2**](#_Toc404837924)

[2.1□□实验目的 1](#_Toc404837921)

[2.2□□实验内容 1](#_Toc404837922)

[2.3□□实验小结 1](#_Toc404837923)

[**3□□□函数与程序结构实验 3**](#_Toc404837929)

[3.1□□实验目的 1](#_Toc404837921)

[3.2□□实验内容 1](#_Toc404837922)

[3.3□□实验小结 1](#_Toc404837923)

[**4□□□编译预处理实验 4**](#_Toc404837934)

[4.1□□实验目的 1](#_Toc404837921)

[4.2□□实验内容 1](#_Toc404837922)

[4.3□□实验小结 1](#_Toc404837923)

[**5□□□数组实验 5**](#_Toc404837938)

[5.1□□实验目的 1](#_Toc404837921)

[5.2□□实验内容 1](#_Toc404837922)

[5.3□□实验小结 1](#_Toc404837923)

[**6□□□指针实验 6**](#_Toc404837943)

[6.1□□实验目的 1](#_Toc404837921)

[6.2□□实验内容 1](#_Toc404837922)

[6.3□□实验小结 1](#_Toc404837923)

[**7□□□结构与联合实验 7**](#_Toc404837948)

[7.1□□实验目的 1](#_Toc404837921)

[7.2□□实验内容 1](#_Toc404837922)

[7.3□□实验小结 1](#_Toc404837923)

[**8□□□文件实验 8**](#_Toc404837953)

[8.1□□实验目的 1](#_Toc404837921)

[8.2□□实验内容 1](#_Toc404837922)

[8.3□□实验小结 1](#_Toc404837923)

[**参考文献 9**](#_Toc404837957)

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

#include<stdio.h>

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

int main(void)

{

char \*s1, \*s2, \*s3;

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);

s3 = strcopy(s1, 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);

}

**解答：**

（1）错误修改：

①第五行的\*s1,\*s2应改为s1[100],s2[100]

②函数strcopy中应定义char \*t0 = t;

返回值应为 t0

（2）错误修改后运行结果：

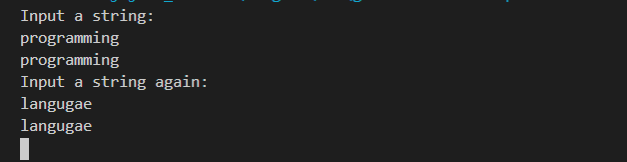


图6.2.1运行结果示意图

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。

#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-j-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]);

return 0;

}

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：

代码如下：

#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]);

    return 0;

}

运行结果图：

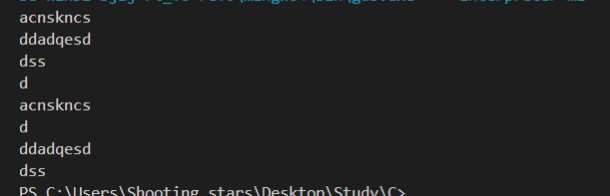


图6.2.2.1运行结果示意图

②：

代码如下：

#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 \*\*s0 = s;

    char \*temp;

    int i, j;

for(i=0; i<size-1; i++)

{

    s = s0;

    for (j=0; j<size-i-1; j++)

        {

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

            {

                temp = \*s;

                \*s = \*(s+1);

                \*(s+1) = temp;

            }

            s++;

        }

}

}

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]);

    return 0;

}

运行结果：

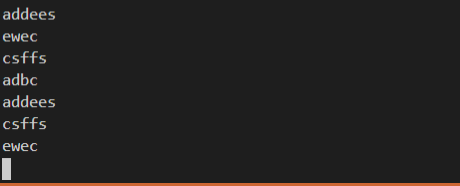


图6.2.2.2运行结果示意图

（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 = strok; break;

case 4: p = 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;

}

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：**

代码如下：

# include<stdio.h>

# include<string.h>

char \*mystrcpy (char \*, char\*);

char \*mystrcat (char \*, char\*);

char \*mystrtok (char \*, char\*);

int main (void)

{

    char \*(\*p)(char \*, 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 = mystrcpy;   break;

        case 2: p = mystrcat;   break;

        case 3: p = mystrtok;   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;

}

char \*mystrcpy (char \*s, char\*t)

{

    char \*s0;

    s0 = s;

    while(\*t != '\0')

    \*s++ = \*t++;

    \*s='\0';

    return s0;

}

char \*mystrcat(char \*s, char\*t)

{

    char \*s0 = s;

    while (\*s != '\0')

    s++;

    while(\*t != '\0')

    \*s++ = \*t++;

    \*s = '\0';

    return s0;

}

char \*mystrtok(char \*s, char \*t)

{

    char \*ps = s, \*pt, \*pc;

    while(\*ps != '\0')

    {

        for(pt = t, pc = ps; \*pt != '\0'&& \*pt== \*pc; pt++, pc++);

        if(\*pt == '\0')

        {

            \*ps = '\0';

            break;

        }

        ps++;

    }

    return s;

}

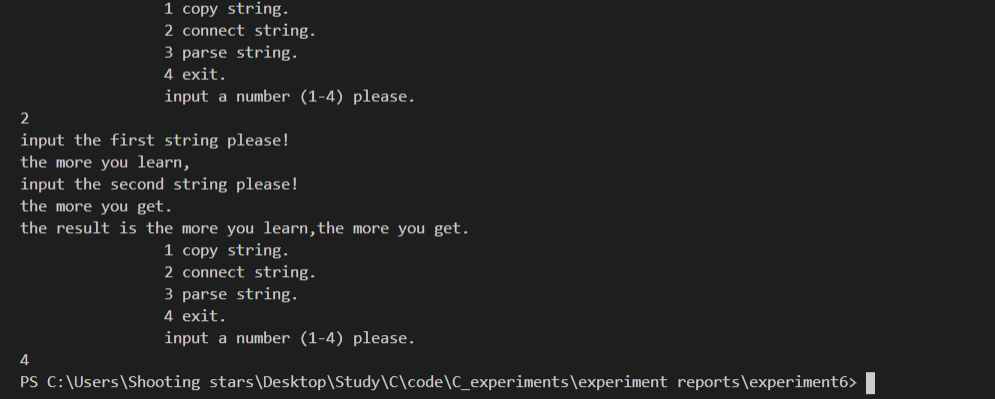
运行结果如下：  


图6.2.2.2运行结果示意图

②：代码如下：

# include<stdio.h>

# include<string.h>

char \*mystrcpy (char \*, char\*);

char \*mystrcat (char \*, char\*);

char \*mystrtok (char \*, char\*);

int main (void)

{

    char \*(\*p)(char \*, char\*);

    char \*(\*s[3])(char \*, char \*);

    s[0] = mystrcpy,s[1] = mystrcat,s[2] = mystrtok;

    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)

        p = s[choice-1];

    else

        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;

}

char \*mystrcpy (char \*s, char\*t)

{

    char \*s0;

    s0 = s;

    while(\*t != '\0')

    \*s++ = \*t++;

    \*s='\0';

    return s0;

}

char \*mystrcat(char \*s, char\*t)

{

    char \*s0 = s;

    while (\*s != '\0')

    s++;

    while(\*t != '\0')

    \*s++ = \*t++;

    \*s = '\0';

    return s0;

}

char \*mystrtok(char \*s, char \*t)

{

    char \*ps = s, \*pt, \*pc;

    while(\*ps != '\0')

    {

        for(pt = t, pc = ps; \*pt != '\0'&& \*pt== \*pc; pt++, pc++);

        if(\*pt == '\0')

        {

            \*ps = '\0';

            break;

        }

        ps++;

    }

    return s;

}

运行结果：

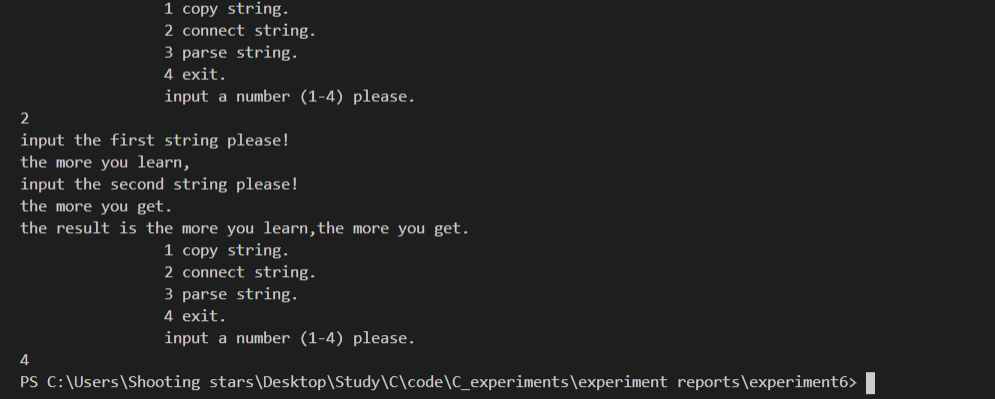


图6.2.2.3运行结果示意图

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.

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

printf("%s\n",strcpy(a,b));

}

char \*strcpy(char \*s,char \*t)

{

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

;

return (s);

}

解答：

1. 进入strcpy时，s为\b随机值，返回main时，s为\0;
2. 函数strcpy改为：

char \*strcpy(char \*s,char \*t)

{

    char \*s0 = s;

    while(\*s++=\*t++);

    return (s0);

}

4、编程设计题

（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：

代码如下：

#include <stdio.h>

int main()

{

    char s[9];

    char \*p = s;

    int a,i;

    long b;

    scanf("%d", &b);

    for(i = 0; i<8; i++)

    {

        a = b>>4\*(7-i) & 0xF;

        if(a>9)

        \*p = a-10+'A';

        else

        \*p = a+'0';

        p++;

    }

    p = '\0';

    printf("%s", s);

    return 0;

}

运行结果示意图：

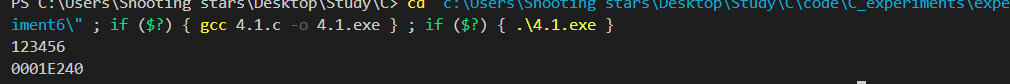


图6.4.1运行结果示意图

（2）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：

代码如下：

#include <stdio.h>

int n,m;

void circle(int [][n], int [][m]);

int main()

{

int i,j;

scanf("%d%d", &n, &m);

int a[n][m],b[m][n];

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

for(j = 0; j<m; j++)

scanf("%d", &a[i][j]);

circle(b, a);

for(j = 0; j<m; j++)

{

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

{

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

if(i <n-1)

printf(" ");

}

if(j <m-1)

printf("\n");

}

return 0;

}

void circle(int b[m][n], int a[n][m])

{

int i,j;

for(j = 0; j<m; j++)

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

b[j][i] = a[i][m-j-1];

}

运行结果：

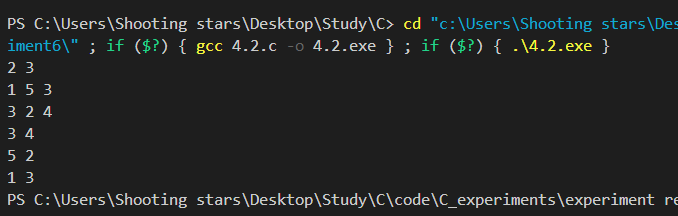


图6.4.2运行结果示意图

（3）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：

代码如下：

#include <stdio.h>

void delete(char \*t);

int main()

{

    int n,i;

    scanf("%d", &n);

    getchar();

    char \*p[n],s[n][80];

    for(i = 0; i<n; i++)

    {

        gets(s[i]);

        p[i] = s[i];

    }

    for(i = 0; i<n; i++)

    delete(p[i]);

    for(i = 0; i<n; i++)

    printf("%s\n", p[i]);

    return 0;

}

void delete(char \*t)

{

    char \*t0 = t;

    int i = 0;

    while(\*t==' '||\*t == '\t')

    i++,t++;

    while (\*(t0+i)!= '\0')

    {

        \*t0 = \*(t0+i);

        t0++;

    }

    \*t0 = '\0';

}

运行结果：

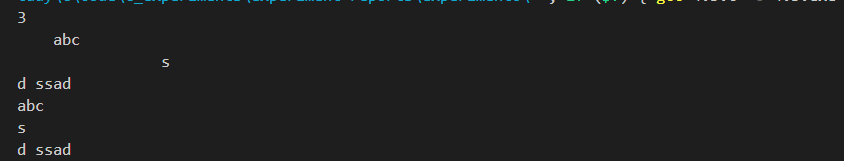


图6.4.3运行结果示意图

（4）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：

代码如下：

#include<stdio.h>

#include<assert.h>

typedef void def(void);

def f0,f1,f2,f3,f4,f5,f6,f7;

def scheduler;

void execute(void(\*p[50])(void), int n);

int main()

{

    scheduler();

    return 0;

}

void scheduler(void)

{

    void (\*p[50])(void);

    char s[50];

    char \*ps = s;

    int i = 0,flag = 1;

    scanf("%s", s);

    while(\*ps != '\0')

    {

        switch (\*ps)

        {

        case '0': p[i] = f0;break;

        case '1': p[i] = f1;break;

        case '2': p[i] = f2;break;

        case '3': p[i] = f3;break;

        case '4': p[i] = f4;break;

        case '5': p[i] = f5;break;

        case '6': p[i] = f6;break;

        case '7': p[i] = f7;break;

        default:printf("Error!");

        flag = 0;

        break;

        }

        assert(flag);

        i++;

        ps++;

    }

    execute(p, i);

}

void execute(void (\*p[50])(void), int n)

{

    int i;

    for(i = 0; i<n; i++)

    (p[i])();

}

void f0()

{

    printf("task0 is called!\n");

}

void f1()

{

    printf("task1 is called!\n");

}

void f2()

{

    printf("task2 is called!\n");

}

void f3()

{

    printf("task3 is called!\n");

}

void f4()

{

    printf("task4 is called!\n");

}

void f5()

{

    printf("task5 is called!\n");

}

void f6()

{

    printf("task6 is called!\n");

}

void f7()

{

    printf("task7 is called!\n");

}

运行结果：

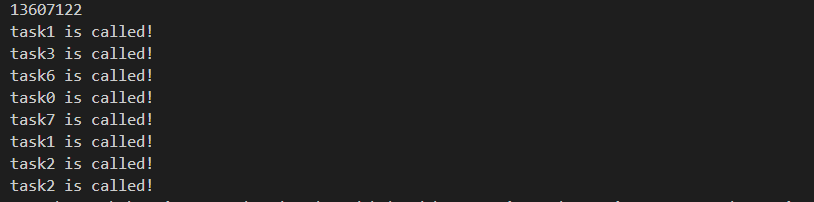


图6.4.4运行结果示意图

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 + 98765432109876543210.0123456789

解答：

代码如下：

#include <stdio.h>

int main()

{

char a[100],b[50];

char \*p;

int \*t,n,i,len1,len2,\*t1,\*t2,temp;

int c1[30]={0}, c2[20]= {0}, d1[30] = {0}, d2[20] = {0};

int len[4] ={0};

gets(a);

p = a;

t = c1;

while(\*p != '.')

{

if(\*p >= '0' &&\*p <= '9')

{

\*t++ = \*p++ - '0';

len[0] ++;

}

else p++;

}

t =c2;

while(\*p != '+')

{

if(\*p >= '0' &&\*p <= '9')

{

\*t++ = \*p++ - '0';

len[1] ++;

}

else p++;

}

t = d1;

while(\*p != '.')

{

if(\*p >= '0' &&\*p <= '9')

{

\*t++ = \*p++ - '0';

len[2] ++;

}

else p++;

}

t = d2;

while(\*p != '\0')

{

if(\*p >= '0' &&\*p <= '9')

{

\*t++ = \*p++ - '0';

len[3] ++;

}

else p++;

}

//初始化

for(i = 0; i<50;i++)

b[i] = '0';

b[30] = '.';

//小数部分相加,但第三十一位暂时不加

len1 = (len[1] >= len[3]) ? len[1] : len[3];

temp = len1;

t1 = &c2[len1 - 1], t2 = &d2[len1 - 1], p = &b[30+len1];

while(len1 > 1)

{

n = \*t1-- + \*t2-- + \*p - '0';

\*p-- = n%10 +'0';

\*p += n/10;

len1--;

}

//第三十一位相加，进位给个位

n = \*t1 + \*t2 + \*p - '0';

\*p = n %10 +'0';

\*(p-2) += n/10;

//整数部分相加;

len2 = (len[0] >= len[2]) ? len[0] : len[2];

t1 = &c1[len2 - 1], t2 = &d1[len2 - 1], p= &b[29];

while(len2 > 0)

{

n = \*t1-- + \*t2-- +\*p - '0';

\*p-- = n%10 + '0';

\*p += n/10;

len2--;

}

//构造字符串并输出

p = &b[temp + 30];

while(\*p -- =='0');

\*(p+1) = '\0';

p = b;

while(\*p++ == '0');

printf("%s", p);

return 0;

}

运行结果：

图6.5.1运行结果示意图

（2）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strcmp、strstr等函数名。

解答：

代码如下：

#include <stdio.h>

#include <string.h>

int main()

{

char \*s = "asdabcfg";

char \*t = "abc";

char \*ret1;

char \*ret2;

char \*(\*p[2])(const char \*, const char \*);

p[0] = strcmp;

p[1] = strstr;

ret1= p[0](s,t);

ret2 = p[1](s,t);

printf("%s", ret2);

return 0;

}

6.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在本次实验中遇到了常量区数据不能修改的问题，在对字符串进行赋值时不能使用=，而应该使用strcpy函数进行操作，还遇到了动态分配内存的问题，在编程时尽量使用函数对代码进行封装，这样可以使得代码更简洁，可读性，实用性更高。

参考文献

[1] 曹计昌,卢萍,李开. C语言程序设计,北京： 科学出版社,2013

[2] 李开,卢萍,曹计昌. C语言实验与课程设计, 北京：科学出版社,2011