

实验 6 指针实验

一、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。

二、实验题目及要求

1. 源程序改错题

下面程序是否存在错误？如果存在，原因是什么？如果存在错误，要求在计算机上对这个例子程序进行调试修改，使之能够正确执行。

```
1. #include "stdio.h"
2. void main(void)
3. {
4. float *p;
5. scanf("%f", p);
6. printf("%f\n", *p);
7. }
```

解答：

(1) 错误修改：

1) 第 4 行使用了未初始化的局部变量“p”，正确形式为：

```
float *p;
float q;
p = &q;
```

(2) 错误修改后运行结果：

```
C:\Users\10334\Documents\Untitled1.exe
3.2
3.200000
Process returned 9 (0x9)   execution time : 0.700 s
Press any key to continue.

微软拼音 半 :
```

2. 源程序完善、修改、替换题

(1) 下面的程序通过函数指针和菜单选择来调用字符串拷贝函数或字符串连接函数，请在下划线处填写合适的表达式、语句、或代码片段来完善该程序。

```
#include "stdio.h"
#include "string.h"
void main(void)
{
    char * (*p)(char *, const char *);
    char a[80], b[80], c[160], *result=c;
    int choice, i;
    do{
        printf("\t\t1 copy string.\n");
        printf("\t\t2 connect string.\n");
        printf("\t\t3 exit.\n");
        printf("\t\tinput a number (1-3) please!\n");
        scanf("%d", &choice);
    }while(choice<1 || choice>5);
    switch(choice) {
    case 1:
        p=strcpy;
        break;
    case 2:
        p=strcat;
        break;
    case 3:
        goto down;
    }
    getchar();
    printf("input the first string please!\n");
```

```

i=0;
gets(a);
printf("input the second string please!\n");
i=0;
gets(b);
result=___p___(a,b);
printf("the result is %s\n",result);
down:
;
}

```

运行结果:

```

C:\Users\10334\Documents\Untitled1.exe
1 copy string.
2 connect string.
3 exit.
input a number (1-3) please!
2
input the first string please!
adfaad hdj
input the second string please!
uiuto hj
the result is adfaad hdjuiuto hj
Process returned 33 (0x21)   execution time : 13.948 s
Press any key to continue.

```

(2) 为了使程序不受 scanf、getchar、gets 等函数输入后回车符的影响, 请修改第 (1) 题程序, 按要求输出下面结果: ((输入) 表示该数据是键盘输入数据)

```

1 copy string.
2 connect string.
3 exit.
input a number (1-3) please!

```

2 (输入)

```

input the first string please!
the more you learn, (输入)
input the second string please!
the more you get. (输入)
the result is the more you learn,the more you get.

```

解答:

(1): 原程序清单:

```

#include "stdio.h"
#include "string.h"
void main(void)
{
    char * (*p)(char *, const char *);

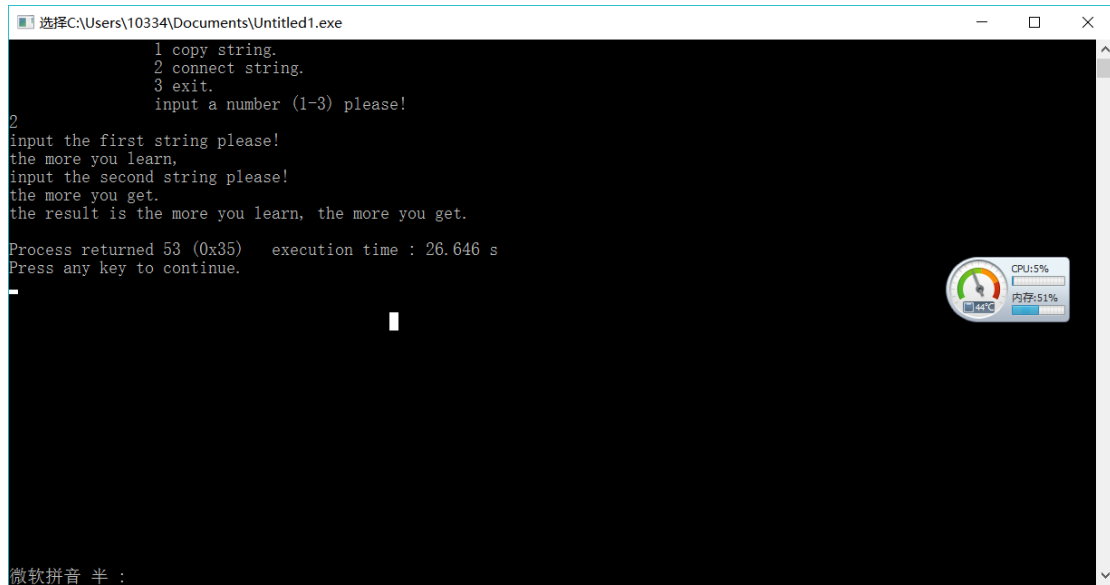
```

```

char a[80], b[80], c[160], *result = c;
int choice, i;
do {
    printf("\t\t1 copy string.\n");
    printf("\t\t2 connect string.\n");
    printf("\t\t3 exit.\n");
    printf("\t\tinput a number (1-3) please!\n");
    scanf("%d", &choice);
} while (choice<1 || choice>5);
switch (choice) {
case 1:
    p = strcpy;
    break;
case 2:
    p = strcat;
    break;
case 3:
    goto down;
}
getchar();
printf("input the first string please!\n");
i = 0;
gets(a);
printf("input the second string please!\n");
i = 0;
gets(b);
result = p(a, b);
printf("the result is %s\n", result);
down:
    ;
}

```

(2): 运行结果:



3. 跟踪调试题

```
#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);
}
```

(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.

(3) 选做：由于 watch 窗口中只显示 s 所指串的值，不显示 s 中存储的地址值，怎样才能观察到 s 值的变化呢？

解答：

(1)：进入时，s=0x004ff7a4；返回 main 时，s=0x004ff7c1。

(2)：

1)：程序清单：

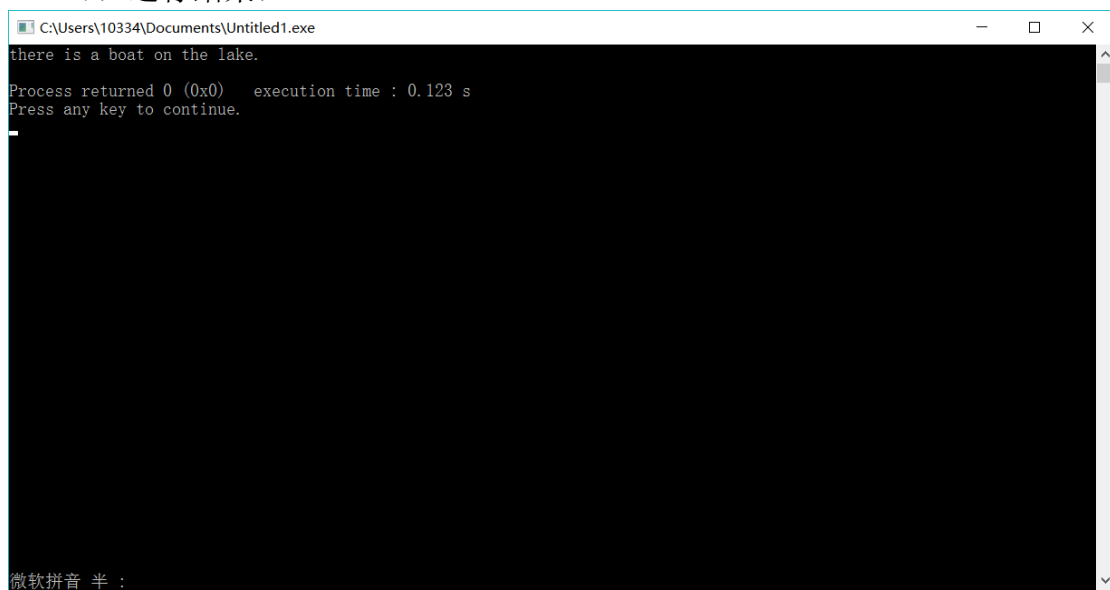
```
#include "stdio.h"
char *strcpy(char *, char *);
void main(void)
```

```

{
    char a[20], b[60] = "there is a boat on the lake.";
    printf("%s\n", strcpy(a, b));
}
char *strcpy(char *s, char *t)
{
    char * p = s;
    while (*s++ = *t++)
        ;
    return (p);
}

```

2): 运行结果:

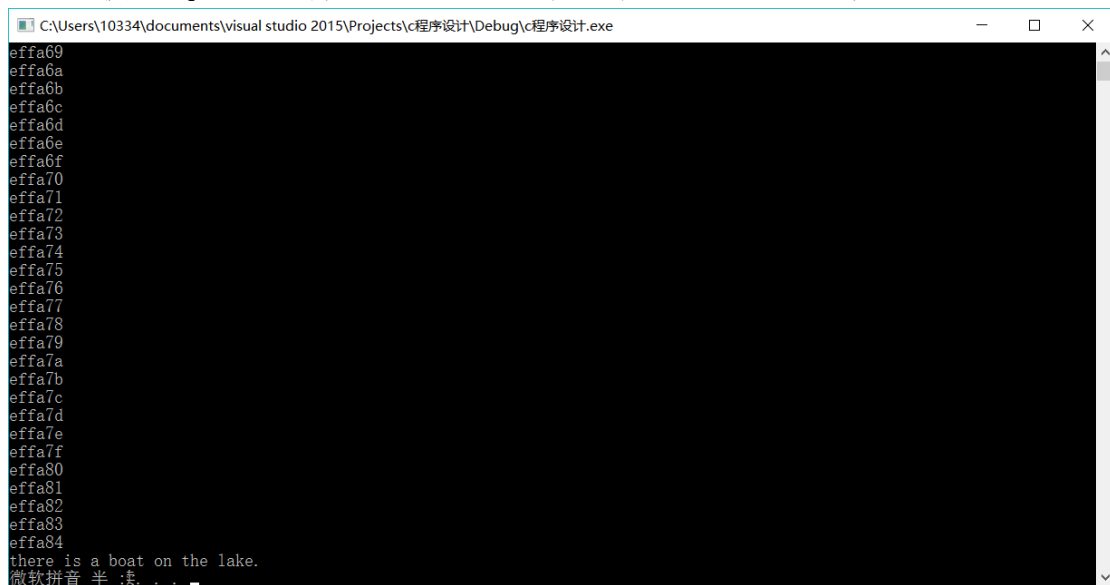


```

C:\Users\10334\Documents\Untitled1.exe
there is a boat on the lake.
Process returned 0 (0x0)   execution time : 0.123 s
Press any key to continue.

```

(3) 使用 printf 将 s 的地址打印出来。Printf("%x",s), 运行结果:



```

C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
effa69
effa6a
effa6b
effa6c
effa6d
effa6e
effa6f
effa70
effa71
effa72
effa73
effa74
effa75
effa76
effa77
effa78
effa79
effa7a
effa7b
effa7c
effa7d
effa7e
effa7f
effa80
effa81
effa82
effa83
effa84
there is a boat on the lake.
微软拼音 半 :

```

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 位并以数字字符的形式进行显示。

解答：

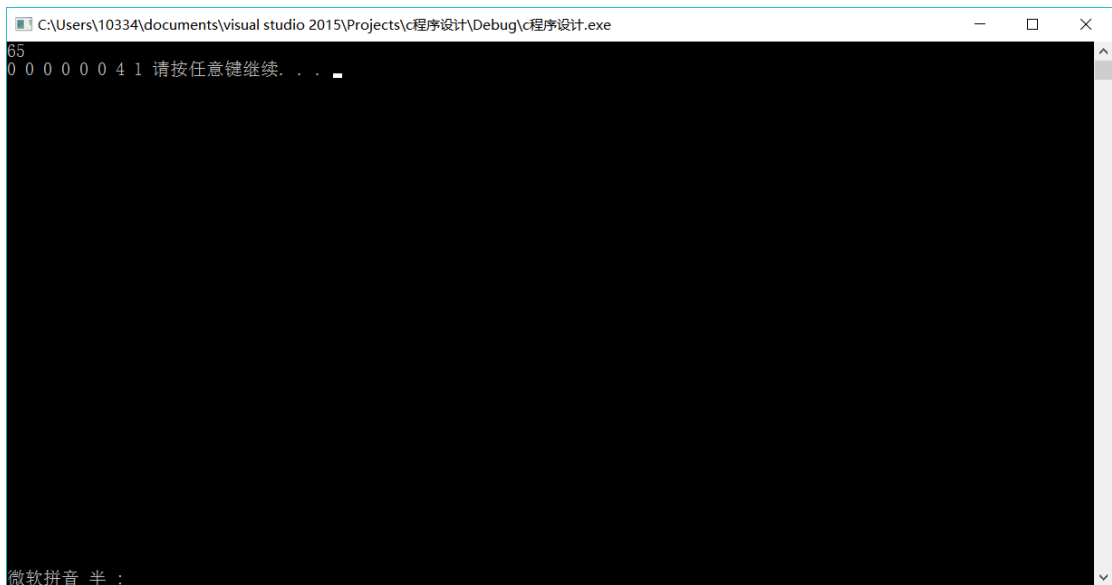
1) 解题思路：

- a) scanf 读取输入的数，记为 num
- b) 构造最高字节的高 4 位为 1 的长整型逻辑尺 (0xf000 0000)；
- c) Num 和 mask 使用&运算符运算可以取出高字节的高 4 位，存储
- d) 将取出来高 4 位右移 28 位得到目标数，进行输出。
- e) 将 num 左移 4 位重新赋值给 num，重复 b) 步骤。
- f) 以上循环 8 次，将 32 位的 long 型按每 4 位依次取出。

2) 程序清单：

```
#include<stdio.h>
int main(void)
{
    long num;
    scanf("%d", &num);
    long mask = 0xf0000000;
    for (int i = 1; i <=8; i++)
    {
        printf("%d ", (mask&num) >> 28);
        num = num << 4;
    }
    system("pause");
}
```

3) 运行结果：



(2) 利用大小为 n 的指针数组指向用 gets 函数输入的 n 行，每行不超过 80 个字符。编写一个函数，它将每一行中连续的多个空格字符压缩为一个空格字符。

在调用函数中输出压缩空格后的各行，空行不予输出。

解答：

1) 解题思路：

- a) 创建指针数组* p[100],用来保存输入的数据。
- b) N 代表有 n 行输出，当 n 为 0 时退出循环。
- c) 对于一给定 n，使用 for 循环处理 n 行输入，每循环一次为 p[i]分配 80 个字节的空间（每行输入不超过 80 个）
- d) 将 fgets 读取的内容存入 p[i]指向的缓冲区中，将字符串的首地址赋值给 p[i]
- e) for 循环 n 次处理 n 行输入
- f) 再次 for 循环 n 次输出处理后的数据，处理数据使用 space_trim() 函数。
- g) 函数接受存储字符串的地址，函数中定义两个指针 pt1,pt2，均指向接收的字符串，使用 while (*pt1) 遍历字符串，*pt2++=*pt1++将 pt1 指向的值赋值给*pt2，并且使用 if 语句判断*pt1 是否为空格，如果是的话使用 while(*pt1==' ')跳过剩下的空格。
- h) 输出之后使用 free() 释放已经分配的动态存储空间。

2) 程序清单：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX_STR 80
char * space_trim(char * c);
int main(void)
{
    char * p[100]; //创建指针数组，共 100 个
    int n, i;
    scanf("%d", &n);
    getchar();
    while (n) //当 n 为 0 时结束程序
    {
        for (i = 0; i < n; i++)
        {
            p[i] = (char *)malloc(MAX_STR * sizeof(char)); //为指针指派 MAX_STR 个字节的缓冲区
            fgets(p[i], MAX_STR, stdin); //将 fgets 读取的内容存入 p[i] 指向的缓冲区中，将字符串的首地址赋值给 p[i]
        }
        for (i = 0; i < n; i++)
        {
            printf("%s", space_trim(p[i])); //输出处理后的字符串
            free(p[i]);
        }
        putchar('\n');
    }
}
```



```

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

    }
    system("pause");
}
char * space_trim(char * c)
{
    char * pt = c;
    char * first = c;
    int i,out=1;
    for (i = 0; i < strlen(c)-1&&out; i++)
        if (*(pt + i) != ' ')
            out=0;//如果字符串 c 从头到尾不全是空格或者换行的话就处
理, c 置为空串;
    if (out) *c = '\0';
    //printf("****%d\n",out);
    /*空格处理*/
    while (*pt!='\0')
    {
        if (*pt == ' ')
        {
            *c++ = *pt++;
            while (*pt == ' ')
                pt++;
        }
        *c++ = *pt++;
    }
    *c = '\0';

    return first;
}

```

3) 运行结果:

```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
1
gafga fadsf          fasdf  fasd
gafga fadsf fasdf fasd
```

- (3) 设某个班有 N 个学生，每个学生修了 M 门课程（用 `#define` 定义 N 、 M ）。输入 M 门课程名称，然后依次输入 N 个学生中每个学生所修的 M 门课程的成绩并且都存放到相应的数组中。编写下列函数：
- a. 计算每个学生各门课程平均成绩；
 - b. 计算全班每门课程的平均成绩；
 - c. 分别统计低于全班各门课程平均成绩的人数；
 - d. 分别统计全班各门课程不及格的人数和 90 分以上（含 90 分）的人数。

在调用函数中输出上面各函数的计算结果。（要求都用指针操作，不得使用下标操作。）

解答：

1) 解题思路：

- a) 使用 `define` 定义 M （科目个数） N （学生个数），方便修改函数
- b) `Score` 二维数组，行存储同一个学生的各科成绩，列存储同一个科目的各学生成绩。
- c) `Subject` 数组记录各个科目名称。
- d) `Name` 数组记录学生姓名。
- e) 第一次 `for` 循环读取 M 个科目的名称。
- f) 第二个 `for` 循环读取学生信息，每一次循环读取该学生姓名以及他的 M 个科目的成绩。
- g) `double average_everyone(int * score, int n);` 函数计算每个人的平均成绩。
 - i. `score` 为分数，`n` 为科目数。
 - ii. 调用该函数时，将该学生的第一个成绩的地址作为参数传递，由于同一个学生的成绩时连续存储的，因此循环相加 `n` 次可以得到该学生的总成绩，除 `n` 就是该学生的平均成绩。
- h) `double average_subject(int * x, int n);` 计算每个科目的平均成绩
 - i. 调用该函数时，将第一个学生的某一科目地址作为第一个参数传递，学生个数作为第二个参数，由于同一个科目成绩位于同一列中，因此该科目的第一个成绩（即第一个学生的该科目成绩）向

后移动 $n * \text{sizeof}(\text{double})$ 个字节（假设成绩为 double 类型）就能访问该科目的第二个成绩

ii. 依次访问同一个科目的各个学生的成绩，相加再除就能得到该科目的平均成绩，将此成绩在返回 main 函数后保存在创建的 average 数组中

i) `int less_than_average(int * x, double average, int n);` 统计低于平均成绩的人数

i. 类似 `average_everyone()` 函数，依次访问每个学生的各科成绩，再同 average 比较

ii. 小于 average 计数器 `count++`。

iii. 返回 `count`。

j) `int perfect_student(int * x, int n);` 统计优秀学生的人数

i. 类似于 `less_than_average()` 函数，同 90 比较，使用 `count` 计数

ii. 返回 `count`

2) 程序清单：

```
#include<stdio.h>
#define M 5//M 个科目
#define N 5//N 个学生
double average_everyone(int * score, int n);
double average_subject(int * x, int n);
int less_than_average(int * x, double average, int n);
int fail_student(int * x, int n);
int perfect_student(int * x, int n);
int main(void)
{
    int score[N][M]; //分数
    char subject[M]; //科目
    char name[N][100]; //姓名
    int i, j;
    double average[M];
    /*读取科目*/
    for (i = 0; i < M; i++)
        scanf("%c", &subject[i]);

    /*读取每个人的姓名和其分数*/
    for (i = 0; i < N; i++)
    {
        scanf("%s", name[i]);
        for (j = 0; j < M; j++)
        {
            scanf("%d", &score[i][j]);
        }
    }
}
```

```

    /*每个人的平均成绩*/
    for (i = 0; i < N; i++)
    {
        printf("Average score of %s is %.2lf\n", name[i],
average_everyone(score[i], M));
    }
    /*每个科目的平均成绩*/
    for (i = 0; i < M; i++)
    {
        average[i] = average_subject(&score[0][i], M);
        printf("Average score of %c is %.2lf\n", subject[i],
average[i]);
    }
    /*统计每个科目低于其平均成绩的人数*/
    for (i = 0; i < M; i++)
    {
        printf("Number of students lower than avg of %c is %d\n",
subject[i], less_than_average(&score[0][i], average[i], N));
    }
    /*统计每个科目不及格的人数*/
    for (i = 0; i < M; i++)
    {
        printf("Number of students %c fail is %d\n", subject[i],
fail_student(&score[0][i], N));
    }
    /*统计每个科目的优秀人数*/
    for (i = 0; i < M; i++)
    {
        printf("Number of students %c perfect is %d\n", subject[i],
perfect_student(&score[0][i], N));
    }
    //system("pause");
}
double average_everyone(int * score, int n)
{
    int i;
    double average = 0;

    for (i = 0; i < n; i++)
    {
        average += *(score + i);
        //printf("%.2lf***\n", average);
    }
    return average / n;
}

```

```

}
double average_subject(int * x, int n)
{
    int i;
    double average = 0;
    for (i = 0; i < n; i++)
    {
        average += *(x + N*i);
    }
    return average / n;
}
int less_than_average(int * x, double average, int n)
{
    int count = 0, i;
    for (i = 0; i < n; i++)
        if (*(x + N*i) <= average)
            count++;
    return count;
}
int fail_student(int * x, int n)
{
    int count = 0, i;
    for (i = 0; i < n; i++)
        if (*(x + N*i) < 60)
            count++;
    return count;
}
int perfect_student(int * x, int n)
{
    int count = 0, i;
    for (i = 0; i < n; i++)
        if (*(x + N*i) >= 90)
            count++;
    return count;
}

```

3) 运行结果:

```
C:\Users\10334\documents\visual studio 2015\Projects\c程序设计\Debug\c程序设计.exe
A B C D E
Zhang
87 88 77 87 95
Li
88 98 100 48 75
Wang
85 68 95 47 59
Han
86 89 75 85 88
Gan
87 68 87 89 100
Average score of Zhang is 86.80
Average score of Li is 81.80
Average score of Wang is 70.80
Average score of Han is 84.60
Average score of Gan is 86.20
Average score of A is 86.60
Average score of B is 82.20
Average score of C is 86.80
Average score of D is 71.20
Average score of E is 83.40
Number of students lower than avg of A is 2
Number of students lower than avg of B is 2
Number of students lower than avg of C is 2
Number of students lower than avg of D is 2
Number of students lower than avg of E is 2
Number of students A fail is 0
Number of students B fail is 0
Number of students C fail is 0
Number of students D fail is 2
Number of students E fail is 1
Number of students A perfect is 0
Number of students B perfect is 1
Number of students C perfect is 2
Number of students D perfect is 0
微软拼音 半 :dents E perfect is 2
请按任意键继续. . .
```

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

(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。

三. 实验总结:

编写 C 语言程序应熟练掌握指针的说明, 赋值, 使用; 熟悉指向数组指针的使用; 同时还要掌握指针函数、函数指针以及指针数组的用法。掌握带有参数的 main 函数的用法。带参数的 main 函数提供了一种向程序传输命令行参数的途径。在所传递的两个参数中, 第一个整型形参表示命令行中字符串的个数, 另一个则是指向命令行中各字符的字符指针数组。带参数 main 函数的运用提供了命令行可选参数, 是程序编写、问题的解决更加多样化。在本次实验过程中, 自己在指针数组应用这方面还不是很熟练, 在用指针表示一维和多维数组时总是不能一次性得到正确的结果, 今后在这方面还有待加强, 还要多加训练。在函数指针这方面, 我有时还是不能准确用指针定义函数的形参, 经常出错, 在修改程序上花费了好多时间。如果能准确高效而且熟练的运用指针, 那将成为编写程序的有力工具! 我们现在还都是初学者, 还要努力, 多加练习。要做到能够熟练运用指针!