

# 第六次作业参考答案

钟文杰      2025.11.12

作业反馈：这次作业大家写的都很好。

## 作业 13

1. 编写一个函数，利用指向字符串的指针，实现 `strlen()` 的功能，即计算一个字符串的长度。（可以先不用指针传递参数，直接写出核心代码即可）

参考代码：

//思路：设置一个指针从字符串开头一直往后检查，遇到第一个'\0'停下

```
#include<stdio.h>
int _strlen(char* str){
    char *tail=str;
    while(tail-str<101){
        if(*tail=='\0') return tail-str;
        tail++;
    }
    return -1;
}
int main(){
    char str[105]={0};
    scanf("%s",str);
    int len=_strlen(str);
    if(len==-1){
        puts("超出处理能力！");
        return -1;
    }
    printf("%d",len);
    return 0;
}
```

## 作业 14

1. 利用指针，编写一个程序，求两个向量的乘积。

参考代码：

//思路：按内积定义计算就好

```
#include<stdio.h>
int main(){
    int n=0,a[15]={0},b[15]={0},*p=a,*q=b,res=0;
    scanf("%d",&n);
    if(n>10){
        puts("超出处理能力！");
        return -1;
    }
    for(int i=0;i<n;i++) scanf("%d",&a[i]);
    for(int i=0;i<n;i++) scanf("%d",&b[i]);
    for(int i=0;i<n;i++) res+=(*p++)*(q++);
    printf("%d",res);
    return 0;
}
```

2. 利用指针，将数组 a 中的  $n$  个整数按相反顺序存放，并输出对换后的结果。（可以用函数，传递必须的参数；也可更简单点，直接用主函数实现）

参考代码：

//思路：设置两个变量，分别指向数组的开头和结尾

//逐位向中间移动，每次移动前先交换二者指向的变量

```
#include<stdio.h>
int main(){
    int n=0,a[15]={0};
    scanf("%d",&n);
    if(n>10){
        puts("超出处理能力！");
    }
    for(int i=0;i<n;i++) scanf("%d",&a[i]);
    for(int *p=a,*q=a+n-1;p<q;p++,q--){
        int tmp=*p;
        *p=*q;
    }
}
```

```

        *q=tmp;
    }
    for(int i=0;i<n;i++) printf("%d ",a[i]);
    return 0;
}

```

3. 编程，分别用 inas 和 outas 作为输入模块函数名和输出模块函数名，对一个整型三行四列的二维数组输入输出，利用指向二维数组元素的指针实现。

**参考代码：**

//思路：注意二维数组的指针的传递方式

```

#include<stdio.h>
void inas(int* a,int m,int n){
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            scanf("%d",a+i*n+j);
        }
    }
    return;
}
void outas(int* a,int m,int n){
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            printf("%d ",*(a+i*n+j));
        }
        printf("\n");
    }
    return;
}
int main(){
    int a[5][5]={0};
    inas(*a,3,4);
    outas(*a,3,4);
    return 0;
}

```

**note:** 指针与二维数组相关的内容不在咱们的课程内容里，考试不会考到。但是二维数组本身是在课程范围里的！