

计算机第九次实验报告

韦璐 PB16000702

学习目录：

1. 指针变量的定义，赋值和运算
2. 指针变量与一维数组
3. 指针变量与字符串
4. 指针变量做函数参数
5. 返回值为指针型函数的定义
6. 指针数组定义与含义
7. 指向二维数组中的元素以及指向二维数组中的一维数组的指针变量的定义和引用
8. 指向指针的指针的定义和引用
9. 命令行参数的定义和含义
10. 指向函数的指针变量的定义，赋值与引用

要点：

1. 地址，指针和数据访问方式

在计算机中，内存是一个具有连续编码的空间，每个存储单元或者字节都有一个唯一的固定的编号，这个编号称为地址。不同的数据类型占据不同字节的存储空间，而每个字节都有一个地址，一般每个数据的首字节地址称为该数据的地址。

- (1) 凡存放在内存中的程序和数据都有一个确定的地址。一个变量的地址成为该变量的指针。用来存放一个变量地址的变量称为指针变量。简单地说，指针就是地址，指针变量就是用来存放地址的变量。
- (2) 通过变量名来直接引用（存取）变量的方式成为直接引用的方式。将变量的地址，存放在一个指针变量中，然后通过存放变量地址的指针变量来引用（存取）变量的方式称为间接引用方式。

2. 指针运算符“&”和“*”

- (1) 取地址符“&”：其操作数可以是各种类型的简单变量、数组元素、结构成员，不能用于表达式，常量和寄存器变量，是表述变量地址的工具。属单目运算符。
- (2) 指针运算符（或者间接访问运算符）“*”，属单目运算符。

其作用如下：1. 定义说明指针变量，如“int i, j, k, *i_pointer;”当指针运算符出现在定义语句中时，说明其后的标识符为存放地址的变量，系统为其分配相应的存储单元。

- (3) 注意：“*”要求其操作数具有指针意义的值。如“k=*i_pointer;”。

3. 指针变量的定义

指针变量必须先定义后引用，在定义中必须指出它所指向对象的数据类型。一旦说明后，就不能随意指向别的不同类型的变量。即将一个变量的地址赋给指针变量时，该变量的类型必须与指针变量的指向类型一致。

指针变量定义的一般方式：

类型标识符 *指针变量名，……；

定义指针变量时可以初始化。例如：

```
int a, *p=&a;
```

4. 指针变量的赋值

- (1) 指针变量赋值常用方法：

1. 使用取地址运算符“&”获取某同类型变量或数组元素的地址，以“指针变量名=&变量名”形式赋值。例如：int a,*p;p=&a;
2. 利用同类型指针变量赋值。例如：int a,*p,*q;p=&a;q=p;

3. 利用特殊的地址符号常量 NULL (等于零) 给地址变量赋空值。例如: `int a,*p;p=NULL;`
4. 取地址运算符“&”的运算对象必须是一个变量。
5. 指针变量是用来存放地址的, 不要给它赋常数值。
6. 在指针变量没有指向确定的存储单元前, 不要对它所指的对象赋值。例如: “`int p;scanf("%d",&p);`”和“`int p;*p=100;`”都是错误的。

5. 指针与数组

指针与数组两者关系密切, 通过指针可方便地存取数组元素。

(1) 数组的指针表示:

1. 数组名: 代表该数组存储单元的起始地址, 可称为数组的指针, 是地址常量。
2. 数组元素: 相当于变量, 代表值, 有固定的存储单元及相应地址, 可通过“&”取地址符得到其地址。对元素的访问类似于变量, 同样可以通过指向数组元素的指针找到所需的元素。数组元素的常用引用方法没有: `int a[10],*p;pa=a;`

`l` 为 `a` 数组内有效下标范围内某一确定序号, 可以有:

1. 通过数组名所代表的地址引用数组元素, `*(a+i)`
2. 通过指针引用数组元素
3. 通过带下标的指针引用数组元素

(3) 如果指针指向数组中某一个确定元素时, 按照定义有: 指针名字+1 指向下一个元素, +i 指向 `i` 个以后的元素。

(4) 指针可以用地址赋值, 可以向后面移动, 但是数组名字不可以。

指针的赋值可以赋值地址, 也可以取地址赋值数值。

(5) `*p++`: 这里 `++` 是后置的方式, 就是先取到数值然后指针指向下一个地址。

6. 指针变量做函数参数

指针变量做函数参数的时候, 改变指针和改变指针所指向的变量的值效果是不同的。

6. 数组名, 指针和函数参数

采用数组名和指针作为函数的参数, 可通过传地址的方式, 实现形参和实参共享存储单元达到数据双向传递的目的。当一个指针变量指向一维数组的第一个元素或者等于等于一维数组名时, 数组名和指针变量的含义相同, 都表示数组的首地址。所以在函数中地址的传递也可以使用指向数组首地址的指针变量, 即实参和形参使用数组名时可以使用指针变量替换。这样实参和形参可以有四种对应组合方式。

形参和实参都用数组名

形参和实参都用指针变量

实参用数组名, 形参用指针变量

实参为指针变量, 形参为数组名

- (1) 一个指针变量可以指向数组中的任何元素, 此时指向该数组元素的指针变量同样可以作为函数参数, 产地的是指向的数组元素的地址。
- (2) 实参指针变量必须有确定的地址, 即必须事先指向一个可使用的存储空间, 否则, 可能由于局部变量的初值的不确定性, 导致用户程序对重要数据区或系统文件的读写, 造成严重的不可预料的后果。

7. 多维数组与指针

(1) 地址的计算与表示:

由于 C 语言按行优先规则存储数组元素, 因此可以把二维数组看做是一维数组的集合, 即二维数组是这样的一个特殊的一维数组, 它的元素是一个一维数组。

(2) 以二维数组为例说明注意点

指针赋值的时候运算符的两边的地址单位要一致。

当二维数组的名字+1 之后就代表了下一行的地址

P 是指向整型存储单元的指针，相当于以元素为单位，故 p+1 就是下移一个元素位置。若要通过 p 访问二维数组

二维数组取某个地址的值的时候可以加星号

用行和列的观点解释多维数组的概念，当数组为三维的时候，可以在行和列的基础上增加层或者页的概念。

- (2) 指针变量作形参以接受通过实参数组名传递来的地址时，可以采用以下常见的形式：

用指向元素的指针变量（单位列）

用指向一维数组的指针变量

- (3) 注意区分指向元素的指针与指向一维数组的指针：指向元素的指针变量，即当指针变量的地址单位等于只带一维下标的二维数组时，它的定义赋值引用与指向一位数组元素的指针变量形式相同。

8. 字符数组和指针

给字符指针初始化的步骤就是将字符串常数的起始地址赋给字符指针变量，使得字符指针指向该字符串。

9. 指针运算

算数 关系

10. 指针数组

在不在*p 周围加（）决定了这玩意儿是否是指针数组

用途：数据排序，只用交换指向字母的指针

11. 返回值为指针型数据的函数

返回数据的存储单元地址或者一个数据集合的存储单元的起始地址

14. 函数指针

就是指向函数的指针

指针函数和函数指针都是在后面加（）区别是前面*p 前有没有加（）

范例 1

题目：排列五个字符串，赋初值和输出都在 main 函数中进行

实验代码：

```
#include<stdio.h>
```

```
#include<string.h>
```

```
main(){
```

```
    void sort(char * s[],int n);
```

```
    static char * str[5]={"basic","fortran","cobol","pascal","c"};
```

```
    int i;
```

```
    sort(str,5);
```

```
    printf("new sequence of string is:\n");
```

```
    for(i=0;i<5;i++)
```

```
        printf("%s\n",str[i]);
```

```
}
```

```
void sort(char * s[],int n){
```

```
    char *temp;int i,j,k;
```

```
    for(i=0;i<n-1;i++){
```

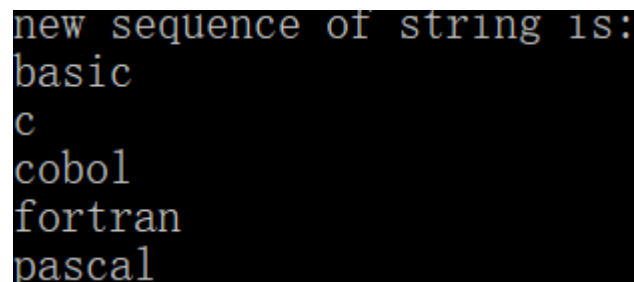
```
        k=i;
```

```

        for(j=i+1;j<n;j++)
        if(strcmp(s[k],s[j])>0) k=j;
        if(k!=i){
            temp=s[i];
            s[i]=s[k];
            s[k]=temp;
        }
    }
}

```

实验结果：



```

new sequence of string is:
basic
c
cobol
fortran
pascal

```

范例二

题目：

写一个程序，给出一个一维数组的元素值，先后四次调用相关函数，求各个元素值的和，下标为奇数的元素之和，各个元素的平均值，最大元素的值。

实验代码：

```

#include<stdio.h>
#define N 12
main(){
    static float a[]={1.5,3.8,5.6,7.8,91.6,1.61,13.3,15.0,17.5,19.9,21.7,23.0};
    float arr_add(float arr[],int n);
    float odd_add(float *p,int n);
    float arr_ave(float *p,int n);
    float arr_max(float a[],int n);
    void process(float *p,float (* fun)(),int n);
    int n=N;
    printf("the sum of %d elements is:",n);
    process(a,arr_add,n);
    printf("the sum of odd elements is:");
    process(a,odd_add,n);
    printf("the average of %d elements is:",n);
    process(a,arr_ave,n);
    printf("the maximum of %d elements is:",n);
    process(a,arr_max,n);
    return;
}
float arr_add(float arr[],int n){
    int i;

```

```

    float sum=0;
    for(i=0;i<n;i++)
        sum=sum+arr[i];
    return(sum);
}
float odd_add(float *p,int n){
    int i;float sum=0;for(i=0;i<n;i=i+2,p=p+2)
        sum=sum+*p;
    return(sum);
}
float arr_ave(float *p,int n){
    int i;float sum=0,ave;
    for(i=0;i<n;i++){
        sum=sum+p[i];
    }
    ave=sum/n;
}
float arr_max(float a[],int n){
    int i=0;float temp=a[i++];
    for(;i<n;i++)
        if(temp<a[i])
            temp=a[i];
    return(temp);
}
void process(float *p,float(*fun)(),int n){
    float result;
    result;
    result=(*fun)(p,n);printf("%8.2f\n",result);
}

```

实验结果：

```

the sum of 12 elements is:  222.31
the sum of odd elements is: 151.20
the average of 12 elements is: 12.00
the maxinum of 12 elements is: 91.60

```

习题

1. 题目：采用冒泡法实现对 n 个整数按递减次序的排序

实验代码：

```

#include<stdio.h>
int main()
{
    int a[10],i,j;
    void sort(int *);//这里是声明函数，这个函数的形参是指针
}

```

```

        for(i=0;i<10;i++)
            scanf("%d",&a[i]);//这里输入数组的元素值
        printf("\nthe original data is :");
        for(i=0;i<10;i++) printf("%d ",a[i]);//这里输出你输入的数组进行确认
        sort(a);//这里用函数，实参是一维数组的首地址
        printf("\nthe result data is :");//输出结果
        for(i=0;i<10;i++)
            printf("%d ",a[i]);//输出你排序好以后的结果
        return 0;
    }
    void sort(int *p)//这里是函数的定义，这个函数的参数是指针
    {
        int i,j,t;
        for(i=0;i<9;i++)
            for(j=0;j<9-i;j++)
                if(p[j]<p[j+1])
                    {t=p[j];p[j]=p[j+1];p[j+1]=t;}//使用冒泡排序法进行排序
    }
}

```

实验结果：

```

22 5 6 49385 5 3 5 6 7 5
the original data is :22 5 6 49385 5 3 5 6 7 5
the result data is :49385 22 7 6 6 5 5 5 5 3

```

调试分析：

无

结果分析：

无

第二题：

题目：采用选择法实现对 n 个整数按递减次序的排序

实验代码：

```

#include<stdio.h>
main()
{
    int i[10]={},c[10]={};
    int a=0,b=0,temp,min;//初始化
    printf("请输入十个整数：");
    for(;a<10;a++)
        scanf("%d",&i[a]);//输入我要排序的数字
    for(a=0;a<9;a++)
    {
        min=a;//查找最小值
    }
}

```

```

        for(b=a+1;b<10;b++)
        {
            if(i[min]>i[b])
            {
                min=b;
            }
        }
        temp=i[min];
        i[min]=i[a];
        i[a]=temp;
    }
    printf("从小到大排序: ");
    for(a=0;a<10;a++)
        printf("%d ",i[a]);
    printf("\n");
    return 0;
}

```

实验结果:

请输入十个整数: 1 2 3 5 4 7 6 9 8 7
从小到大排序: 1 2 3 4 5 6 7 7 8 9

调试分析: 选择法是比较大小以后就进行交换

实验总结: 记住选择法

第三题

题目: 将数组 a 中 n 个整数按相反顺序存放, 并输出对换后的结果。

实验代码:

```

#include<stdio.h>
#include<stdlib.h>
int N;
int exchange(int *p);//这里声明我们定义了一个以指针为参数的函数
int main()
{
    int i,a[100]={0},*pointer=a;//这里进行初始化
    printf("要输入几个数字: ");
    scanf("%d",&N);//输入的数据
    printf("输入你需要的数: ");
    for(i=0;i<N;i++)
        scanf("%d",pointer++);//这里进行数据输入, 输入完以后指针就指向最后一个数
    了, 所以我们这里要进行将
    pointer=a;//指针重新放到第一个的操作
    exchange(pointer);//然后输入地址进行交换
    pointer=a;
    printf("倒序结果: ");
    for(i=0;i<N;i++) printf("%d ",*pointer++);//然后我们这里输出排序的结果, 并对

```

这个指针进行取值以后再将它指向下一个

```
        system("pause");
    }
    int exchange(int *p)//交换函数的定义
    {
        int i,temp,time=(N-1)/2,*pp;
        pp=p+N-1;//此时 p 指向第一个元素，而 pp 指向最后一个
        for(i=0;i<=time;i++,p++,pp--)//如果是奇数个中间的不用换了，假如是偶数就
没关系
        {
            temp=*p;
            *p=*pp;
            *pp=temp;//这里就是对前半部分的指针所指向的内容和后半部分所指向
的内容进行交换，我们交换的时候进行了取值
        }
        return 0;
    }
}
```

实验结果：

要输入几个数字：5	要输入几个数字：4
输入你需要的数：1 2 3 4 5	输入你需要的数：5 6 7 8
倒序结果：5 4 3 2 1 请按任意键继续.	倒序结果：8 7 6 5 请按任意键继续.

实验总结：就是使用指针的时候一定要抓住指针的本质，指针只负责指向某个地址。

第四题：

题目：

编一个函数实现一个 $n \times n$ 的矩阵转置。在程序的主函数中用 scanf 函数输入以下矩阵元素，将数组名作为函数实参，函数调用后在主函数中输出已转置矩阵。

实验代码：

```
#include <stdio.h>
#define n 4//因为我们要输入的是一个 4 阶的矩阵
void transpose(int a[n][n],int b[n][n]);//声明转置的函数
int main()
{
    int a[n][n],b[n][n],i,j;//声明我们的矩阵和我们要使用的参数
    printf("请输入一个 4×4 的矩阵:\n");//加强程序的可读性
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);//进行矩阵的输入
    printf("\n 转置后的矩阵为:\n");
    transpose(a,b);//进行矩阵的转置
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            printf("%4d",b[i][j]);//输出转置以后的矩阵
        putchar(10);//进行换行
    }
}
```



```

    }
    return 0;
}

void transpose(int a[n][n],int b[n][n])
{
    int i,j;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            b[j][i]=a[i][j];//进行矩阵的转置
}

```

实验结果：



```

请输入一个4×4的矩阵:
1  2  3  4
5  6  7  8
9  10 11 12
13 14 15 16

转置后的矩阵为:
1  5  9 13
2  6 10 14
3  7 11 15
4  8 12 16
Press any key to continue

```

实验总结：无

第五题

题目：写一个函数，求一个字符串的长度。在主函数中输入字符串，并输出其长度。

实验代码：

```

#include <stdio.h>

// 计算字符串 str 长度，并返回
int lenstr(const char* str)//定义一个函数以字符型指针为形参
{
    int i = -1;
    while (str[++i]); // while 循环在 str[++i] 为 '\0' 时结束
    return i;//返回个数
}

int main()
{
    char p[1000];
    printf("input a string:");
    gets(p);//使用函数接收我们要得到的函数
    printf("string len: %d\n", lenstr(p));//得到我们要得到的结果
}

```

```
        return 0;
    }
}
```

实验结果：

```
input a string:aioq
string len: 4
```

实验总结：可以直接在 printf 中引用函数作为参量

第六题

题目：编写函数实现两个字符串的连接

实验代码：

```
#include <stdio.h>

#include <string.h>

int main()
{
    char s1[30];

    char s2[10];//这里定义两个字符串数组

    int len1,len2;//字符串 1、2 的长度

    int i;

    printf("请输入字符串 1: \n");

    gets(s1);//这里进行两个字符串的输入

    printf("请输入字符串 2: \n");

    gets(s2);

    len1=strlen(s1);//这里求长度

    len2=strlen(s2);

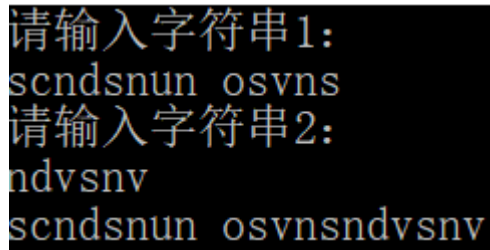
    for(i=0;i<=len2;i++)
```

```
s1[len1+i]=s2[i]; //然后将第二个的字符放到第一个里面
```

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

```
}
```

实验结果:



```
请输入字符串1:
scndsnun osvns
请输入字符串2:
ndvsnv
scndsnun osvnsndvsnv
```

第八题

题目: 编写函数实现两个字符串的比较

实验代码:

```
#include <stdio.h>
#include <stdlib.h>
int strcmp(char *p1, char *p2); //定义 strcmp 函数
int main()
{
    char s1[20], s2[20];
    printf("Please enter s1 string: "); //输入字符串 s1
    gets(s1);
    printf("Please enter s2 string: "); //输入字符串 s2
    gets(s2);
    printf("Result: %d\n", strcmp(s1, s2)); //输出返回值
    system("pause");
    return 0;
}
//strcmp 函数
int strcmp(char *p1, char *p2) //定义函数, 函数的形参是指针
{
    int r, t; //定义两个变量
    for (; *p1==*p2&&(*p1!='\0' || *p2!='\0'); p1++, p2++); //当两个指针
    指向的东西相等, 并且都没有结束的时候, 就指向下一个内容
    r=*p1-*p2; //当上面的条件不满足的时候, 就将他们的 ascii 码相减
    r==0 ? t=0 : t=*p1-*p2;
    return t;
}
```

```
Please enter s1 string: wge
Please enter s2 string: eggush
Result: 18
请按任意键继续. . .
```

实验结果：

实验总结：指针的用处很大，好好使用会有很多巧妙的用法。

第九题

题目：编写函数，其功能是在 float 类型一维数组中查找最大值，最小值并将它们返回到调用程序并输出。要求：采用形实参数结合传递的方法实现。

实验代码：#include <stdio.h>

```
void fun(float a[],float *mx,float *mi)//这里定义函数，函数的形参有数组的地址和两个指针
{
```

```
    int i=0;//初始化
```

```
    for (i=0;i<10;i++)
```

```
    {
```

```
        if (i==0)
```

```
        {
```

```
            *mx = *mi = a[i];//这里是为了防止出现错误
```

```
        }else
```

```
        {
```

```
            if (*mx<a[i])
```

```
            {
```

```
                *mx = a[i];//这里对每一个数进行遍历，然后将大的数赋值
```

```
            }
```

```
            if (*mi >a[i])
```

```
            {
```

```
                *mi = a[i];//同理
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

```
main()
```

```
{
```

```
    int i=0,n=0;
```

```
    float a[10]={0},maxf=0,minf=0;//定义数组和最大数最小数并进行初始化
```

```
    scanf("%d",&n);//输入
```

```
    for (i=0;i<10;i++)
```

```
    {
```

```
        scanf("%f",&a[i]);//输入数组的元素
```

```
    }
```

```
    fun(a,&maxf,&minf);//然后调用函数进行排序，并且这里传递的参数是这两个常数的地
```

址

```
printf("MAXF[%f] MINF[%f]\n",maxf,minf);//这里进行结果的输出
}
```

```
3
2 3 6.7 4 4 3 2 6 7.9 9
MAXF[9.000000] MINF[2.000000]
```

实验结果：

11

题目：输入 5 个字符串，用指向指针的指针方法按从小到大的顺序排序并输出。

实验代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void sort(char *s[5], int n);          //定义排序函数
int main()
{
    char *str[5], **p;//定义一个指针数组和指向指针的指针
    int i;//定义一个变量
    for (i=0; i<5; str[i++]= (char *)malloc(20*sizeof(char)));          //分配
字符串空间
    for (p=str, printf("Please enter 5 strings: "); p<str+5; scanf("%s", *p++));          //输入 5 个字
字符串，这里可以对地址的大小进行比较
    sort(str, 5); //使用函数对这五个字符串进行比较
    for (p=str, printf("Sort by: "); p<str+5; printf("%s ", *p++));          //输出排序
后的 5 个字符串
    printf("\n");
    return 0;
}
//排序函数
void sort(char *s[5], int n)
{
    int i, j;
    char *t;
    for (i=0; i<n; i++)
        for (j=i+1; j<n; strcmp(s[i], s[j])>0 ? t=s[i], s[i]=s[j], s[j]=t, j++ : j++);
}
```

实验结果：

```
Please enter 5 strings: noewefn hfuiife hfuosh houhf suhuor ooi
Sort by: hfuiife hfuosh houhf noewefn suhuor
```

12

实验题目：利用指针的动态申请空间方式程序,求两个向量的乘积

实验代码：

```
#include<stdio.h>
int DotProduct(int a[],int b[],int len)
```

```

{
    int ret = 0;
    for(int i=0;i<len;i++)
    {
        ret += a[i]*b[i];
    }
    return ret;
}
int main()
{
    int len = 0;//定义并进行初始化
    scanf("%d",&len);//输入维度
    int *a = new int[len];
    int *b = new int[len];
    for(int i=0;i<len;i++) scanf("%d",a+i);//输入 a
    for(int i=0;i<len;i++) scanf("%d",b+i);//输入 b
    printf("%d\n",DotProduct(a,b,len));//点积
    delete[]a;//释放内存
    delete[]b;
    return 0;
}

```

实验结果：

```

2
1 2
3 4
11

```