# 一、实验目的与要求

1. 掌握一维和多维数组的定义和数组元素的引用方法。
2. 了解一维和多维数组初始化等基本算法。
3. 掌握指针的定义方法，指针的操作符和指针的运算。
4. 理解指针与数组、字符串的关系。

# 二、实验内容：数组、指针的使用

### 实验1、随机数排序（）

### 问题的简单描述：用冒泡排序法实现，对用随机函数产生的10个整数按升序排列。

### 2实验代码：

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

void sort1(int s[],int n)

{

int I,j;

int temp;

for(i=9;i>0;i++);

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

if(s[j]>s[j+1])

{

temp=s[j];

s[j]=s[j+1];

s[j+1]=temp;

}

}

main()

{

int I,a[10];

srand((unsigned int )time(NULL));

printf(“随机产生10个整数: \n”);

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

a[i] =rand()%100;

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

printf(“%d “,a[i]);

printf(“ln”);

sort1(a,10);

printf(“输出排序后的10个整数\n”);

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

printf(“%d “,a[i]);

}

### 实验2、杨辉三角的实现（）

**问题描述：编写程序，从键盘输入行数，输出指定行数的杨辉三角形。**

**实验代码：**

#include<stdio.h>

main()

{

int a[50][50],i,j,n;

printf("请输入杨辉三角形的行数: ") ;

scanf( "%d" ,&n);

for(i=1;i<=n;i++)

{

a[i][i]=1;

a[i][1]=1;

}

for(i=3;i<=n;i++)

{

for(j=2;j<=i-1;j++)

a[i][j]=a[i-1][j-1]+a[i-1][j];

}

for(i=1;i<=n;i++)

{

for(j=1;j<=i;j++)

printf("%-4d",a[i][j]);

printf("\n");

}

}

 问题分析：数值要对齐，注意%。文末附有杨辉三角的六种解法，想深入学习的同学可以看看。

### 实验3 字符串反转及字符串连接（）

**问题描述：定义两个字符指针，通过指针移动方式将字符串反转以及将两个字符串连接起来。  
实验代码：**

#include<stdio.h>

char \*reverse(char \*str);

char \*link(char \*str,char \*str2);

int main()

{

char str[30],str1[30],\*str2;

printf("Input Reversing Character String:");

gets(str);

str2=reverse(str);

printf("\nOutput Reversed Character String:");

puts(str2);

printf("Input string1:");

gets(str);

printf("Input string2:");

gets(str1);

str2=link(str,str1);

puts(str2);

return 0;

}

char \*reverse(char \*str)

{

char \*p,\*q,temp;

p=str,q=str;

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

p++;

p--;

while (q<p)

{

temp=\*q;

\*q=\*p;

\*p=temp;

q++;

p--;

}

return str;

}

char \*link(char \*str1,char \*str2)

{

char \*p=str1,\*q=str2;

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

p++;

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

{

\*p=\*q;

q++;

p++;

}

\*p='\0';

return str1;

}

问题分析：

ps：请将该实验内容保存至泛雅平台或者百度网盘里，备写后期实验报告。

# 附：C语言求杨辉三角的6种方法

2016-12-25 15:19:16 https://csdnimg.cn/release/phoenix/template/new_img/articleReadEyes.png87555 https://csdnimg.cn/release/phoenix/template/new_img/tobarCollect.png收藏 40

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

的三角形，其实质是二项式(a+b)的n次方展开后各项的系数排成的三角形，它的特点是左右两边全是1，从第二行起，中间的每一个数是上一行里相邻两个数之和。这个题目常用于程序设计的练习。

下面给出六种不同的解法。

解法一

#include <stdio.h>

main()

{ int i,j,n=0,a[17][17]={0};

while(n<1 || n>16)

{ printf("请输入杨辉三角形的行数:");

scanf("%d",&n);

}

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

a[i][0]=1; /\*第一列全置为一\*/

for(i=1;i<n;i++)

for(j=1;j<=i;j++)

a[i][j]=a[i-1][j-1]+a[i-1][j];/\*每个数是上面两数之和\*/

for(i=0;i<n;i++) /\*输出杨辉三角\*/

{ for(j=0;j<=i;j++)

printf("%5d",a[i][j]);

printf("\n");

}

}

点评：解法一是一般最容易想到的解法，各部分功能独立，程序浅显易懂。

解法二

#include <stdio.h>

main()

{ int i,j,n=0,a[17][17]={1};

while(n<1 || n>16)

{ printf("请输入杨辉三角形的行数:");

scanf("%d",&n);

}

for(i=1;i<n;i++)

{ a[i][0]=1; /\*第一列全置为一\*/

for(j=1;j<=i;j++)

a[i][j]=a[i-1][j-1]+a[i-1][j]; /\*每个数是上面两数之和\*/

}

for(i=0;i<n;i++) /\*输出杨辉三角\*/

{ for(j=0;j<=i;j++)

printf("%5d",a[i][j]);

printf("\n");

}

}

点评：解窢二是在解法一的基础上，把第一列置为1的命令移到下面的双重循环中，减少了一个循环。注意初始化数组的变化。

解法三

#include <stdio.h>

main()

{ int i,j,n=0,a[17][17]={0,1};

while(n<1 || n>16)

{ printf("请输入杨辉三角形的行数:");

scanf("%d",&n);

}

for(i=1;i<=n;i++)

for(j=1;j<=i;j++)

a[i][j]=a[i-1][j-1]+a[i-1][j]; /\*每个数是上面两数之和\*/

for(i=1;i<=n;i++) /\*输出杨辉三角\*/

{ for(j=1;j<=i;j++) printf("%5d",a[i][j]);

printf("\n");

}

}

点评：解法三是在解法一、二的基础上，把第一列置为1的命令去掉了，注意初始化数组的变化。

解法四

#include <stdio.h>

main()

{ int i,j,n=0,a[17][17]={0,1};

while(n<1 || n>16)

{ printf("请输入杨辉三角形的行数:");

scanf("%d",&n);

}

for(i=1;i<=n;i++)

{ for(j=1;j<=i;j++)

{ a[i][j]=a[i-1][j-1]+a[i-1][j]; /\*每个数是上面两数之和\*/

printf("%5d",a[i][j]); /\*输出杨辉三角\*/

}

printf("\n");

}

}

点评：解法四是在解法三的基础上，把计算和打印合并在一个双重循环中。

解法五

#include <stdio.h>

main()

{ int i,j,n=0,a[17]={1},b[17];

while(n<1 || n>16)

{ printf("请输入杨辉三角形的行数:");

scanf("%d",&n);

}

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

{ b[0]=a[0];

for(j=1;j<=i;j++)

b[j]=a[j-1]+a[j]; /\*每个数是上面两数之和\*/

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

/\*输出杨辉三角\*/

{ a[j]=b[j]; /\*把算得的新行赋给a,用于打印和下一次计算\*/

printf("%5d",a[j]);

}

printf("\n");

}

}

点评：解法一到解法四都用了二维数组，占用的空间较多。而解法五只使用了两个一维数组。

解法六

#include <stdio.h>

main()

{ int i,j,n=0,a[17]={0,1},l,r;

while(n<1 || n>16)

{ printf("请输入杨辉三角形的行数:");

scanf("%d",&n);

}

for(i=1;i<=n;i++)

{ l=0;

for(j=1;j<=i;j++)

{ r=a[j];

a[j]=l+r; /\*每个数是上面两数之和\*/

l=r;

printf("%5d",a[j]); /\*输出杨辉三角\*/

}

printf("\n");

}

}

点评：解法六只使用了一个一维数组和两个临时变量。