数组容量必须是常量表达式。

为了系统安全，定义数组时尽量初始化，否则，数组元素的值是不可预测的。

数组的初始化

int a[10]={0};

double a[10] = {0.0};

char c[10] = {0};

以上3个数组的存储单元都被初始化成0，这是C语言提倡的初始化方法。

小结：如果只给数组部分赋值，其他单元编译器会自动赋0，但如果一个单元也不赋值，编译器不会赋0，即值仍是不可预测的。

static int arr[10]；数组分配在静态区，这样的数组会自动初始化0仅且初始化一次

数组变量名是数组的首地址，是一个占4字节的普通指针常量，除了sizeof（数组名）表示数组的长度这一特例。

任何指针都可以当做数组名，进行+-和下标访问，而且不检查是否越界。

一维数组作为函数参数，使用指针作为形参。如int arr[7]；形参是int \*Arr；sizeof（Arr）=4，实参到形参是值传递，形参在函数内操作时不需要被保护，修改形参值值并不会改变实参，且形参完全等效是一个普通的指针，实参仍是数组名。

int getsize(int data[])

{

return sizeof(data);

}

int main()

{

int data[] = { 1, 2, 3, 4 };

int size1 = sizeof(data);

int szie2 = getsize(data);

int \*p = data;

int size3 = sizeof(p);

/\*输出结果：size1 = 16；size2 = size3 = 4\*/

为了提高数组性能，C语言并不检查数组的索引是否越界，所以程序员要自己注意：访问数组时不要越界

二维数组

int matrix[3][10] : 共有3个元素，每个元素是一个还有10个整型的数组，matrix指向第一个数组的首地址。注意区分matrix和指向整数的指针的区别！

matrix+1是指第二个数组

int \*p = matrix 错误

int (\*p)[10] = matrix 正确

int (\*p)[4];//数组指针 p是指针，指向一维数组,每个一维数组有4个int元素。

二维数组的函数形参必须显示第二维：

int matrix [ ][10]  或者 int (\*matrix)[10]

最佳初始化方法

int matrix[ ][3] = { {0,0,0},{1,1,1},{2,2,2} };

字符列表有两种存储方式

1，用矩阵存储即二维数组，行长度等于最长字符的长度，列数等于字符串的个数，数组是定长的。

2，指向字符串常量的指针数组形式存储，指针指向的字符串存储单元是不定长恰适的。

字符串

定义字符串：①char str[ ] = "hello world" ②char \*str = "hello world"

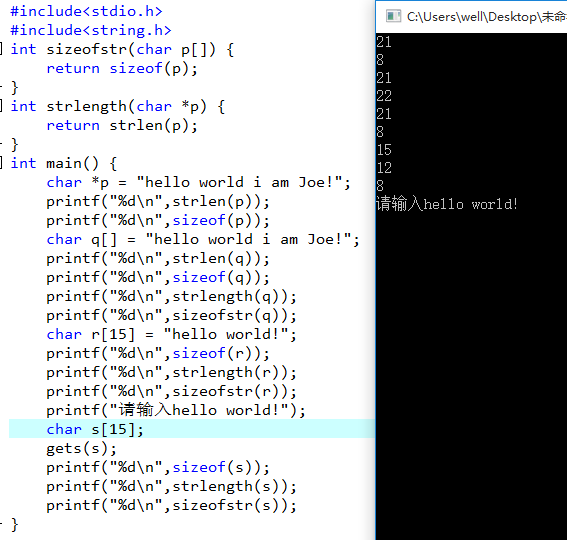
①12个元素的数组，str是数组的首地址，sizeof（str） = 48；

②str是个普通指针变量，指向字符串常量的存储位置，也是首地址，sizeof（str） = 4；

字符串常量存储在内存的一个特定区域，当用方法②创建多个字符串，若有多个字符串是相同的，便只存储一个，多个指针指向这个字符串。

char str1[ ] = "hello world"；char str2[ ] = "hello world"【str1不等于str2】

char \*str1 = "hello world"；char \*str2= "hello world" 【str1等于str2】



/\*键盘输入字符串标准格式

char str[10];

gets(str);

//gets(str)可被scanf(“%s”,&str)代替，区别是，输入的字符串有空格，gets会把空格当字符，scanf却把空格当输入结束符号。

或

char \*p = (char\*)malloc(sizeof(char)\*10);

gets(p);

下面是不对的

char \*str;

gets(str);

\*/

程序中连续使用两个及以上的scanf，要在每个scanf后面加上fflush(stdin);