知识点1【数组的概述】（了解）

知识点2【一维数值数组的定义】（重要）

1、定义数组的步骤：

案例1：定义一个数组 有5个元素 每个元素为int

案例2：定义一个数组 有5个元素 每个元素为int *

案例3：定义一个数组 有5个元素 每个元素为数组，该数组有10个元素每个元素为int

案例4：定义一个数组 有5个元素 每个元素为函数的入口地址，该函数有两个int型形参，int返回值类型

知识点3【一维数值数组的初始化】（重要）

1、全部元素 初始化

2、部分元素 初始化

1、未被初始化的部分 自动补0

2、建议将数组的所有元素初始化为0

3、指定下标初始化(了解)

知识点4【一维数值数组的元素操作】（重要）

1、元素的操作　对元素的读或写

2、键盘给数组元素获取输入

案例1：键盘输入10个int数　求这10个数的最大值和最小值

案例2：键盘输入10个int数　将这10个数 逆置。

案例3：键盘输入10个int数　将这10个数 排序(选择法)

知识点4【二维数值数组概述】（了解）

知识点5【二维数值数组的初始化】（重要）

1、分段初始化

2、连续初始化

知识点6【二维数值数组的元素操作】（重要）

知识点7【一维字符数组】（重要）

1、逐个元素初始化（不推荐）

2、字符串的方式 初始化 一维字符数组（推荐）

3、字符数组 遍历

使用循环方式逐个遍历（逐个字符操作）

4、键盘 获取 字符数组

1、scanf 和 %s获取字符串 遇到空格或回车 结束输入

2、获取带空格的 字符串 gets （建议别用）

知识点8【一维字符的案例】（重要）

1、获取一个字符串 求该字符串的长度（不适用strlen）

2、有两个字符数组str1 str2, str1获取键盘输入，输入后将str1的字符串，拷贝到 str2中（不允许使用strcpy）

3、有以下两个数组str1,str2 将str2的字符串 追加到 str1的尾部

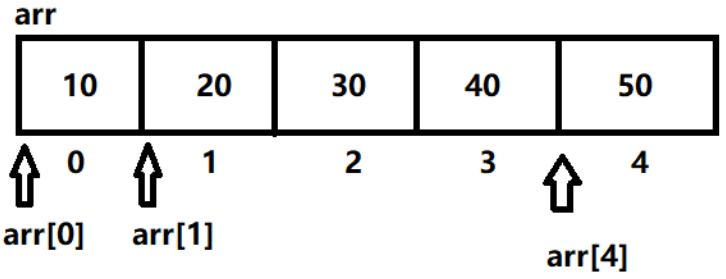
4、从字符串中 查找某个元素（第一次出现的下标）

5、键盘输入第一个字符串str1,输入第二个字符串str2,再输入位置pos,要求将str2字符串插入到字符串str1的pos位置

知识点9【二维字符数组】（了解）

知识点1【数组的概述】（了解）

```
int num1 = 10;
int num2 = 20;
int num3 = 30;
int num4 = 40;
int num5 = 50;
```



用一段连续空间 存放相同类型的变量 这样的容器（结构）叫数组。

数组的下标是从0开始。

加入数组有n个元素：

下标范围：0~n-1

元素范围：arr[0]~arr[n-1]

案例1：以下对int arr[5]的数组元素 访问正确的是___A____

```
A:arr[1] B:arr+1 C:arr[5] D:arr
```

知识点2【一维数值数组的定义】（重要）

1、定义数组的步骤：

数组名 和 [] 表示数组

将数组的个数 放入[]里面

用元素的类型 定义一个变量

从上 往下 替换。

案例1：定义一个数组 有5个元素 每个元素为int

```
int arr[5];
```

案例2：定义一个数组 有5个元素 每个元素为int *

```
int *arr[5];
```

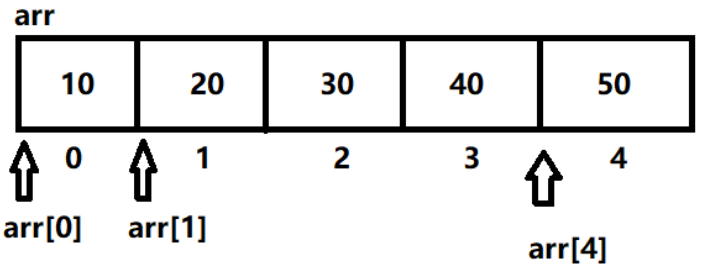
案例3：定义一个数组 有5个元素 每个元素为数组，该数组有10个元素每个元素为int

```
int arr[5][10];
```

案例4：定义一个数组 有5个元素 每个元素为函数的入口地址，该函数有两个int型形参，int返回值类型

```
int (*arr[5])(int, int);
```

int arr[5]; 数组的总大小 == 元素的个数 * 每个元素的大小



数组名作为类型：sizeof(arr) ==数组的总大小

sizeof(arr[0]) ==元素的大小

sizeof(arr)/sizeof(arr[0]) == 元素的个数

不管几维：数值数组 必须逐个元素范围。

```
 7  *       Compiler:  gcc
 8  *         Author:  YOUR NAME (),
 9  *        Company:
10  * ======================================================================*/
11
12
13 #include <stdio.h>
14 void test01()
15 {
16     int arr[5];
17     int n = sizeof(arr)/sizeof(arr[0]);
18
19     printf("数组的总大小=%lu\n", sizeof(arr));
20     printf("数组元素大小=%lu\n", sizeof(arr[0]));
21     printf("数组元素的个数=%lu\n", sizeof(arr)/sizeof(arr[0]));
22
23     int i=0;
24     for(i=0;i<n;i++)
25     {
26         printf("%d ", arr[i]);
27     }
28     printf("\n");
29
30 }
31 int main(int argc, char *argv[])
32 {
33     test01();
34     return 0;
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
数组的总大小=20
数组元素大小=4
数组元素的个数=5
1 0 4196093 0 0
edu@edu:~/work/c/day03$
```

局部数组不初始化内容为 不确定。

知识点3【一维数值数组的初始化】（重要）

1、全部元素 初始化

```
1  int arr[5]={10,20,30,40,50};
```

注意：如果数组的全部元素 都初始化 可以省略[]的数值

如果省略了[]里面数值 数组的元素个数 就由初始化的元素个数确定

```
1  int arr[]={10,20,30,40,50};
```

2、部分元素 初始化

1、未被初始化的部分 自动补0

```
1  int arr[5]={10,20,30};
```

```
void test02()
{
    int arr[5]={10,20,3};
    int n = sizeof(arr)/sizeof(arr[0]);

    int i=0;
    for(i=0;i<n;i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");

}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
10 20 3 0 0
edu@edu:~/work/c/day03$
```

2、建议将数组的所有元素初始化为0

```
1  int arr[5]={0};//将第0个元素初始化为0 其他未被初始化自动补0 推荐
2  int arr[5]={10};//10 0 0 0 0
```

```c
void test02()
{
    int arr[5]={0};
    int n = sizeof(arr)/sizeof(arr[0]);

    int i=0;
    for(i=0;i<n;i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");

}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
0 0 0 0 0
edu@edu:~/work/c/day03$
```

3、指定下标初始化(了解)

```c
int arr[5]={[2]=10, [4]=30};//0 0 10 0 30
```

```c
void test02()
{
    int arr[5]={[2]=10, [4]=30};
    int n = sizeof(arr)/sizeof(arr[0]);

    int i=0;
    for(i=0;i<n;i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");

}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
0 0 10 0 30
edu@edu:~/work/c/day03$
```

知识点4【一维数值数组的元素操作】（重要）

1、元素的操作　对元素的读或写

必须逐个元素操作

数组的每个元素 等价于 变量。　arr[1]　== num

```c
void test03()
{
    int arr[5]={0};
    int n = sizeof(arr)/sizeof(arr[0]);

    //num取值
    printf("%d\n", arr[2]);

    //num赋值 num=10
    arr[2]=10;

    //data = num
    arr[3] = arr[2];

    //num++
    arr[3]++;//arr[3] = arr[3]+1

    int i=0;
    for(i=0;i<n;i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
0
0 0 10 11 0
edu@edu:~/work/c/day03$
```

2、键盘给数组元素获取输入

```c
void test04()
{
    int arr[5] = {0};
    int n = sizeof(arr)/sizeof(arr[0]);

    printf("请输入%d个int数据:", n);
    int i=0;
    for(i=0;i<n; i++)
    {
        scanf("%d", &arr[i]);
    }

    for(i=0;i<n;i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入5个int数据:10 20 304 50 60
10 20 304 50 60
edu@edu:~/work/c/day03$
```

案例1：键盘输入10个int数 求这10个数的最大值和最小值

```c
void test05()
{
    int arr[10]={0};
    int n = sizeof(arr)/sizeof(arr[0]);

    printf("请输入%d个int数据:", n);
    int i=0;
    int max = 0, min = 0;
    for(i=0;i<n;i++)
    {
        scanf("%d", &arr[i]);
        if(i == 0)
        {
            min = max = arr[i];
        }

        //进行比较
        max = max>arr[i]?max:arr[i];
        min = min<arr[i]?min:arr[i];
    }

    printf("max = %d, min = %d\n", max, min);

}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入10个int数据:9 3 1 2 5 6 4  7 8 12
max = 12, min = 1
edu@edu:~/work/c/day03$
```

案例2：键盘输入10个int数 将这10个数 逆置。

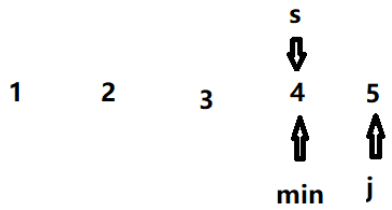
```
1 2 3 4 5 6 7 8 9 10 -----> 10 9 8 7 6 5 4 3 2 1
```

```c
void test05()
{
    int arr[10]={0};
    int n = sizeof(arr)/sizeof(arr[0]);

    printf("请输入%d个int数据:", n);
    int i=0;
    for(i=0;i<n;i++)
    {
        scanf("%d", &arr[i]);
    }

    //逆置
    int j = 0;
    for(i=0,j=n-1; i<j; i++, j--)
    {
        int tmp = arr[i];
        arr[i] = arr[j];
        arr[j] = tmp;
    }

    for(i=0;i<n;i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入10个int数据:1 2 3 4 5 6 7 8 9 0
0 9 8 7 6 5 4 3 2 1
edu@edu:~/work/c/day03$
```

案例3：键盘输入10个int数 将这10个数 排序(选择法)

int arr[5]={3, 5, 2, 4,1}; n = 5



第0轮 初始化s=0,min=s,j=min+1; j<n; j++

第1轮 初始化s=1,min=s,j=min+1; j<n;j++

第2轮 初始化s=2,min=s,j=min+1;j<n;j++

第3轮 初始化s=3,min=s,j=min+1;j<n;j++

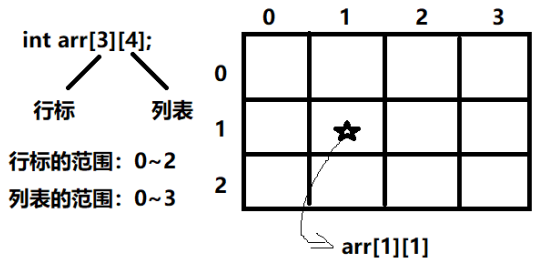
```
for(s=0; s<n-1; s++)
    for(j=min+1; j<n;j++)
        if(arr[min] > arr[j])
            min = j;

if(s != min)
    int tmp=arr[s];  arr[s]=arr[min]; arr[min]=tmp;
```

```c
void test05()
{
 int arr[10]={0};
 int n = sizeof(arr)/sizeof(arr[0]);

 printf("请输入%d个int数据:", n);
 int i=0;
 for(i=0;i<n;i++)
 {
  scanf("%d", &arr[i]);
  }

 //选择法排序
 int s= 0;
 for(s=0;s<n-1; s++)
 {
 int min = s;
 int j = min+1;
 for(; j<n; j++)
 {
 if(arr[min] > arr[j])//从小--->大
 {
 min = j;//纪录最小值的下标
 }
 }
```

```
26
27  //判断假设的最小值下标 和 真实的最小值下标是否 相等 如果不相等 交换
28  if(s != min)
29  {
30  int tmp = arr[s];
31  arr[s] = arr[min];
32  arr[min] = tmp;
33  }
34  }
35
36
37  for(i=0;i<n;i++)
38  {
39  printf("%d ", arr[i]);
40  }
41  printf("\n");
42 }
```

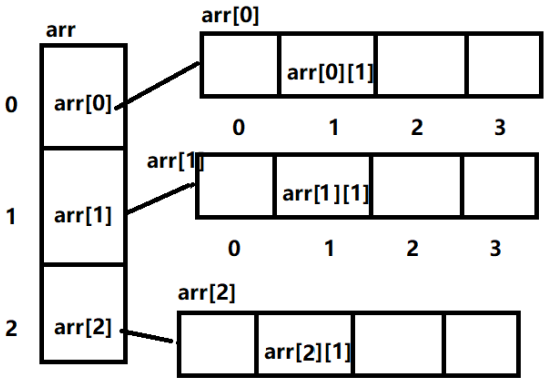
```
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入10个int数据:9 0 2 5 8 7 1 3 4 6
0 1 2 3 4 5 6 7 8 9
edu@edu:~/work/c/day03$
```

知识点4【二维数值数组概述】（了解）



二维数组：数组的数组

二维数组的总大小==行数*列数*每个元素的大小
数组的总大小 == sizeof(arr)
行数：sizeof(arr)/sizeof(arr[0])
列数：sizeof(arr[0])/sizeof(arr[0][0]);



知识点5【二维数值数组的初始化】（重要）

1、分段初始化

```
//完全初始化
int arr[3][4]={ {1,2,3,4}, {5,6,7,8}, {9,10,11,12} };
//如果完全初始化 只能省略行数
int arr[][4]={ {1,2,3,4}, {5,6,7,8}, {9,10,11,12} };

//部分初始化
int arr[3][4]={ {1,2}, {5,6}, {9,10,11} };
```

2、连续初始化

```
/完全初始化
int arr[3][4]={ 1,2,3,4, 5,6,7,8, 9,10,11,12};
//如果完全初始化 只能省略行数
int arr[][4]={1,2,3,4, 5,6,7,8, 9,10,11,12};

//部分初始化
int arr[3][4]={ 1,2, 5,6, 9,10,11};
```

案例1：以下代码的结果是__11__

```
int arr1[3][4]={ {1,2}, {5,6}, {9,10,11} };
int arr2[3][4]={ 1,2, 5,6, 9,10,11};
arr1[1][2] +arr2[1][2] == 11
```

知识点6【二维数值数组的元素操作】（重要）

	语文	数学	化学	物理
老大	56	75	78	89
老二	89	98	76	67
老三	88	99	77	66
老四	67	78	89	90
老五	98	97	96	95

```c
void test06()
{
    int arr[3][4]={0};
    int row = sizeof(arr)/sizeof(arr[0]);
    int col = sizeof(arr[0])/sizeof(arr[0][0]);

    printf("请输入%d个int数据\n", row*col);
    int i=0,j=0;
    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            scanf("%d", &arr[i][j]);
        }
    }

    for(i=0;i<row;i++)
    {
        for(j=0;j<col;j++)
        {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }

}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入12个int数据
1 2 3 4
5 6 7 8
9 10 11 12
1 2 3 4
5 6 7 8
9 10 11 12
edu@edu:~/work/c/day03$
```

案例1：

```c
int arr[5][4]={{56,75,78,89},{89,98,76,67},{88,99,77,66},{67,78,89,90},{98,97,96,95}};
```

```c
void test07()
{
    int arr[5][4]={{56,75,78,89},{89,98,76,67},{88,99,77,66},{67,78,89,90},{98,97,96,95}};
    int row = sizeof(arr)/sizeof(arr[0]);
    int col = sizeof(arr[0])/sizeof(arr[0][0]);
    //定义一个一维数组 存放每个人的平均成绩
    int grade_avg[5]={0};

    int i=0;
    for(i=0;i<row;i++)
    {
        int sum = 0, j = 0;
        for(j=0;j<col; j++)
        {
            sum += arr[i][j];
        }

        grade_avg[i] = sum/col;
    }

    for(i=0;i<row;i++)
    {
        printf("%d ", grade_avg[i]);
    }
    printf("\n");
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
74 82 82 81 96
edu@edu:~/work/c/day03$
```

```c
void test07()
{
    int arr[5][4]={{56,75,78,89},{89,98,76,67},{88,99,77,66},{67,78,89,90},{98,97,96,95}};
    int row = sizeof(arr)/sizeof(arr[0]);
    int col = sizeof(arr[0])/sizeof(arr[0][0]);
    //定义一个一维数组 存放学科的平均成绩
    int avg[4]={0};

    int j = 0;
    for(j=0;j<col;j++)
    {
        int i=0,sum=0;
        for(i=0;i<row;i++)
        {
            sum += arr[i][j];
        }
        avg[j] = sum/row;

    }

    int i=0;
    for(i=0;i<col;i++)
    {
        printf("%d ", avg[i]);
    }
    printf("\n");
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 00_code.c
edu@edu:~/work/c/day03$ ./a.out
79 89 83 81
edu@edu:~/work/c/day03$
```

知识点7【一维字符数组】（重要）

1、逐个元素初始化（不推荐）

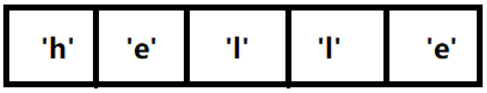
```c
char arr[5]={'h','e','l','l','o'};
```

2、字符串的方式 初始化 一维字符数组（推荐）

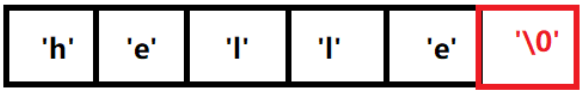
""描述的是字符串， 比如字符串"hello", 编译器会自动在字符串的末尾 添加'\0'字符 作为字符串的结束标记

```c
char arr[6]="hello";
```

char arr[5]={'h','e','l','l','o'};



char arr[6]="hello";



3、字符数组 遍历

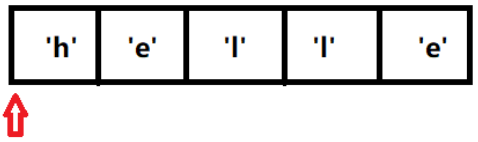
使用循环方式逐个遍历（逐个字符操作）

```
1 char arr[6]="hello";
2 int i=0;
3 for(i=0;i<6;i++)
4 {
5  printf("%c", arr[i]);
6 }
7
8
```

使用%s 直接输出字符串，需要字符串的首元素地址，遇到'\0'才结束。(遍历字符串)

```
1 char arr[6]="hello";
2 printf("%s", arr);
```

char arr[5]={'h','e','l','l','o'};

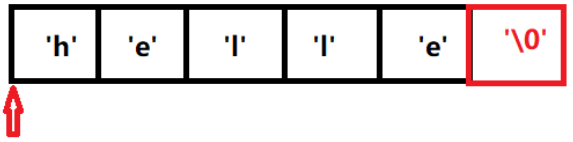


arr

printf("%s", arr);

数组名作为地址 代表的是 首元素地址。

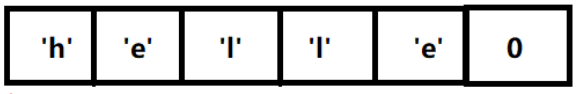
char arr[6]="hello";



arr

printf("%s", arr);

由于逐个元素初始化没有'\0',所以使用%s输出容易访问非法内存



char arr[6]={'h','e','l','l','o'};

printf("%s", arr);

只初始化5个元素 最后一个元素自动补0 =='\0' 所以%s 输出没问题

```c
#include <stdio.h>
#include <string.h>
void test01()
{
    char arr1[]={'h','e','l','l','o'};
    char arr2[]="hello";

    //sizeof测量的是 数组空间总大小
    printf("sizeof(arr1)=%lu\n",sizeof(arr1));//5
    printf("sizeof(arr2)=%lu\n", sizeof(arr2));//6

    //strlen 测量的是字符串的长度（不计算'\0'），遇到'\0'结束长度计算
    printf("strlen(arr1)=%lu\n",strlen(arr1));//不确定 概述数组没有'\0'访问非法内存
    printf("strlen(arr2)=%lu\n",strlen(arr2));//5

    char arr3[]="hello\0world";
    printf("sizeof(arr3)=%lu\n",sizeof(arr3));//12
    printf("strlen(arr3)=%lu\n",strlen(arr3));//5

    // 两个反斜杠 算一个字符
    char arr4[]="\123456\xab\\abc";
    printf("sizeof(arr4)=%lu\n",sizeof(arr4));//10
    printf("strlen(arr4)=%lu\n",strlen(arr4));//9

}
int main(int argc, char *argv[])
{
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
sizeof(arr1)=5
sizeof(arr2)=6
strlen(arr1)=5
strlen(arr2)=5
sizeof(arr3)=12
strlen(arr3)=5
sizeof(arr4)=10
strlen(arr4)=9
edu@edu:~/work/c/day03$
```

4、键盘 获取 字符数组

1、scanf 和 %s获取字符串 遇到空格或回车 结束输入

```c
void test02()
{
    char str[128]="";//推荐的方式

    printf("请输入一个字符串:");
    scanf("%s", str);

    printf("输出结果:%s\n", str);
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
输出结果:hello
edu@edu:~/work/c/day03$
```

2、获取带空格的 字符串 gets （建议别用）

gets可以获取带空格的字符串 但是不会获取回车

```c
void test02()
{
    char str[128]="";//推荐的方式

    printf("请输入一个字符串:");
    gets(str);

    printf("输出结果:##%s##\n", str);
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
输出结果:##hello world##
edu@edu:~/work/c/day03$
```

gets存在的风险：

```c
void test02()
{
    char str[6]="";//推荐的方式

    printf("请输入一个字符串:");
    //gets在获取键盘输入的时候 不会判断目的空间 是否足够 容易造成越界
    gets(str);

    printf("输出结果:##%s##\n", str);
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
输出结果:##hello world##
*** stack smashing detected ***: ./a.out terminated
已放弃 (核心已转储)
edu@edu:~/work/c/day03$
```

3、fgets函数 可以获取带空格的字符串 安全

```c
char *fgets(char *s, int size, FILE *stream)
```

s:表示空间的起始位置

size：表示的能获取的最大字节数size-1

stream:表示设备（stdin标准输入设备（终端））

```c
void test02()
{
    char str[6]="";//推荐的方式

    printf("请输入一个字符串:");
    fgets(str, sizeof(str), stdin);

    printf("输出结果:##%s##\n", str);
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
输出结果:##hello##
edu@edu:~/work/c/day03$
```

fgets 能获取带空格的字符串 遇到回车（达到最空间大值）会结束获取，注意fgets会提取回车

```c
void test02()
{
    char str[128]="";//推荐的方式

    printf("请输入一个字符串:");
    fgets(str, sizeof(str), stdin);

    printf("输出结果:##%s##\n", str);
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
输出结果:##hello world
##
edu@edu:~/work/c/day03$
```

消除获取到的回车

```c
void test02()
{
    char str[128]="";//推荐的方式

    printf("请输入一个字符串:");
    fgets(str, sizeof(str), stdin);
    str[strlen(str)-1]=0;//消除获取到的回车

    printf("输出结果:##%s##\n", str);
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
输出结果:##hello world##
edu@edu:~/work/c/day03$
```

知识点8【一维字符的案例】（重要）

1、获取一个字符串 求该字符串的长度（不适用strlen）

```c
void test03()
{
    char str[128]="";

    printf("请输入一个字符串:");
    fgets(str,sizeof(str),stdin);
    str[strlen(str)-1] = 0;

    //求字符串的长度
    int i=0;
    while(str[i] != '\0')
    {
        i++;
    }
    printf("字符串的长度为:%d\n", i);
}
```

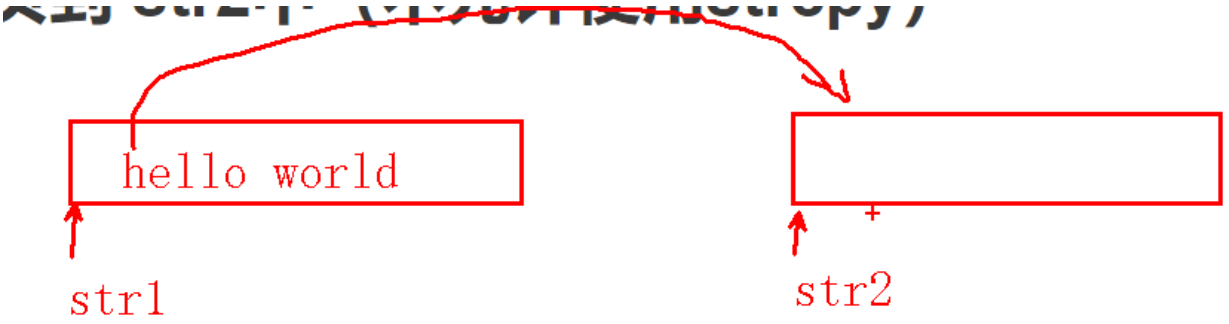
```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
字符串的长度为:11
edu@edu:~/work/c/day03$
```

高級

```c
void test03()
{
    char str[128]="";

    printf("请输入一个字符串:");
    fgets(str,sizeof(str),stdin);
    str[strlen(str)-1] = 0;

    //求字符串的长度
    int i=0;
    while(str[i] && ++i);

    printf("字符串的长度为:%d\n", i);
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
字符串的长度为:11
edu@edu:~/work/c/day03$
```

2、有两个字符数组str1 str2，str1获取键盘输入，输入后将str1的字符串，拷贝到 str2中（不允许使用strcpy）

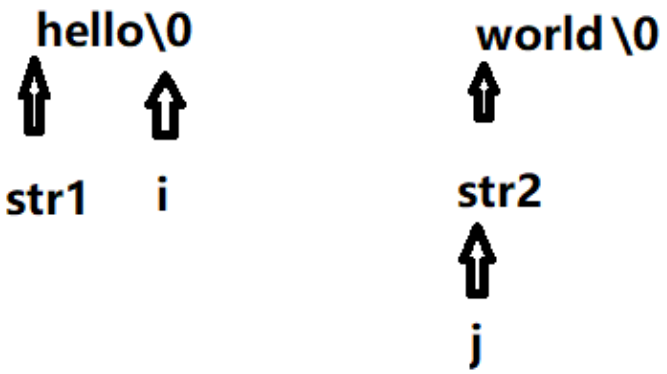



```c
void test03()
{
    char str1[128]="";
    char str2[128]="";

    printf("请输入一个字符串:");
    fgets(str1,sizeof(str1),stdin);
    str1[strlen(str1)-1] = 0;

    //字符串拷贝
    int i=0;
    while(str1[i] != '\0')
    {
        str2[i] = str1[i];
        i++;
    }
    str2[i]='\0';

    printf("str2=##%s##\n", str2);
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
请输入一个字符串:hello world
str2=##hello world##
edu@edu:~/work/c/day03$
```

3、有以下两个数组str1,str2 将str2的字符串 追加到 str1的尾部

```c
char str1[128]="hello";
char str2[128]="world";

str1--->"helloworld"
```



1、将i指向str1的尾部（'\0'下标）

2、循环的将str1[i] = str2[j],当str2[j] == '\0' 结束

```c
void test04()
{
    char str1[128]="hello";
    char str2[128]="world";

    //将i定位到尾部
    int i=0;
    while(str1[i] && ++i);

    //将str2的每个元素 复制到 str1的尾部
    int j=0;
    while(str2[j] != '\0')
    {
        str1[i] = str2[j];
        i++;
        j++;
    }
    str1[i]='\0';

    printf("##%s##\n", str1);
}
```

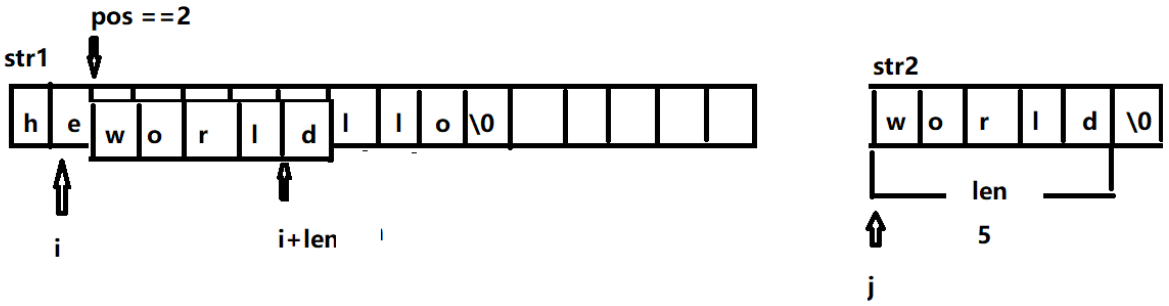
```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
##helloworld##
edu@edu:~/work/c/day03$
```

4、从字符串中 查找某个元素（第一次出现的下标）

```c
void test05()
{
    char str[128]="";

    printf("亲输入一个字符串:");
    fgets(str,sizeof(str),stdin);
    str[strlen(str)-1]=0;

    char ch='\0';
    printf("请输入要查找的字符:");
    ch = getchar();

    //开始查找
    int i=0;
    while(str[i] != '\0')
    {
        if(str[i] == ch)
        {
            printf("找到位置为:%d\n", i);
            break;
        }
        i++;
    }

    if(str[i] == '\0')
    {
        printf("没有找到\n");
    }
}
int main(int argc, char *argv[])
{
    test05();
    return 0;
}
```

```
edu@edu: ~/work/c/day03
edu@edu:~/work/c/day03$ gcc 01_code.c
edu@edu:~/work/c/day03$ ./a.out
亲输入一个字符串:hello world
请输入要查找的字符:o
找到位置为:4
edu@edu:~/work/c/day03$ ./a.out
亲输入一个字符串:hello wrold
请输入要查找的字符:h
找到位置为:0
edu@edu:~/work/c/day03$ ./a.out
亲输入一个字符串:hello world
请输入要查找的字符:H
没有找到
edu@edu:~/work/c/day03$
```

5、键盘输入第一个字符串str1,输入第二个字符串str2,再输入位置pos,要求将str2字符串插入到字符串str1的pos位置



1、计算str2的长度

2、将i定位到str1的尾部

3、str1移动

```
while(i>=pos)
str1[i+len] = str1[i];
i--;
```

4、将str2的每个元素 插入到 pos中

```
while(str2[j] != '\0')
str1[pos] = str2[j ]
pos++
j++
```

```c
void test06()
{

```

```
char str1[128]="";
printf("亲输入一个字符串:");
fgets(str1,sizeof(str1),stdin);
str1[strlen(str1)-1]=0;

char str2[128]="";
 printf("亲输入第二个字符串:");
 fgets(str2,sizeof(str2),stdin);
 str2[strlen(str2)-1]=0;

 int pos=0;
 printf("请输入要插入的位置:");
 scanf("%d", &pos);

 //计算str2的长度
 int len = 0;
 while(str2[len] && ++len);

 //将i定位到str1的尾部 i也是str1的长度
 int i=0;
 while(str1[i] && ++i);

 //判断pos位置是否 合法
 if(pos<0 || pos>i)
 {
 printf("下标%d不无效\n", pos);
 return;//结束函数
 }

 //数据是否溢出
 if(i+len>sizeof(str1))
 {
 printf("插入会越界, 失败\n");
 return;//结束函数
 }

 //在str1移动数据 预留足够的位置
 while(i>=pos)
 {
 str1[i+len] = str1[i];
 i--;
```

```
45  }
46
47  //将str2的字符串 插入到pos位置上
48  int j=0;
49  while(str2[j] != '\0')
50  {
51  str1[pos] = str2[j];
52  pos++;
53  j++;
54  }
55
56
57  printf("插入后的结果：%s\n", str1);
58 }
59 int main(int argc, char *argv[])
60 {
61  test06();
62  return 0;
63 }
```

知识点9【二维字符数组】（了解）

```
1 char str[128]="hello";
2
3 char arr[5][128]={"hello","world", "hehehe", "xixixi", "lalala"};
```

```
void test01()
{
    char arr[5][128]={"hello","world", "hehehe", "xixixi", "lalala"};
    int row = sizeof(arr)/sizeof(arr[0]);
    //遍历二维字符数组
    int i=0;
    for ( i = 0; i < row; i++)
    {
        //arr[i] 代表的是 每一行的第0列的列地址
        printf("%s\n", arr[i]);
    }

}
```

```
edu@edu: ~/work/c/day04
edu@edu:~/work/c/day04$ ./a.out
hello
world
hehehe
xixixi
lalala
edu@edu:~/work/c/day04$
```

获取多个字符串，使用二维字符数组。

```
1 void test01()
2 {
3     char arr[5][128] = {""};
```

```c
    int row = sizeof(arr) / sizeof(arr[0]);
    int col = sizeof(arr[0]) / sizeof(arr[0][0]);

    //获取字符数
    printf("请输入%d个字符串\n", row);
    int i = 0;
     for (i = 0; i < row; i++)
     {
         // scanf("%s", &arr[i][0]);
         scanf("%s", arr[i]);
     }

     printf("-----------------\n");
     //遍历二维字符数组
     for (i = 0; i < row; i++)
     {
         //arr[i] 代表的是 每一行的第0列的列地址
         printf("%s\n", arr[i]);
     }
}
```

```
edu@edu:~/work/c/day04$ sudo gcc 00_code.c
edu@edu:~/work/c/day04$ ./a.out
请输入5个字符串
hehehe xixixi lalala heihei hello
-----------------
hehehe
xixixi
lalala
heihei
hello
edu@edu:~/work/c/day04$
```