任何的传递过程中，
只要不传递地址的值，不传递数组，
都无法在分立的函数中修改主函数的值

主要的内容包括
字符串指针：
字符指针是指指向字符型数据的指针变量
每个字符串在内存中都占用一段连续的内存空间，并有唯一的首地址。因此只要将首地址赋值给字符指针，就可以让字符指针指向一个字符串。
例如使用下面的方式进行定义的处理：
```
char *pat = "hello";
```
或者有：因为字符串也是一个数组，并且其的值也是一个地址的值，所以可以直接进行赋值的处理
也就是定义一个字符串的的地址
```
char *pat;
pat = "hello";
```
字符串在只读的常量存储区中，所以可以修改变量的值，但是不能对pat所指向的单元进行写操作

当然也可以用字符的指针指向字符数组的首地址，然后进行指向的修改即可

有很多的时候的很多的函数都被设计成了有字符串处理函数是不需要返回值的。
because：增加使用时的灵活性，如支持表达式的链式表达式，方便进行操作。
例如
```
char *mylief(char *dstStr,char *scrStr)
char *pat = date;
return pat;
```
其的返回值就是一个字符指针的格式
这样在返回和修改的时候就可以修改主函数中的字符串或者字符指针
从而完成修改的作用和返回修改的作用。

动态数据结构，单链表的格式与形式：
单链表的格式和方式
单链表属于一种动态的数据结构，
动态的内存存储方式

结构图配合指针实现

首先，结构体在声明的时候不能包含本结构体的成员（无法分配内存呀）

然而，声明结构体类型的时候可以包含指向本结构体类型的指针成员。

```
typedef struct lib
{
int data;
LIB *list;
}LIB;
```

链表的每一个元素称为一个节点

由数据域

和指针域

一般是线性链表或者单向链表，访问的时候只能进行顺序访问，不能进行随机访问。

接下来完成一个由单链表构成的增删改查的处理过程

详情请关照数据结构下的算法的体系内容

这里主要讲结构体指针的定义及其初始化

```
LIB *pt;
pt = str;
```

将结构体看做是一种特殊的变量是很好的理解方法

先看两个符号

结构体变量名.成员名

```
str.name  = wang;
```

指向结构体的指针变量名->成员名

```
str->name  = wang;
```

指向结构体数组的指针的形式如下：

```
LIB *pt = stu;

LIB *pt = &stu[0];

LIB *pt ;数组的名称代表的就是数组的首地址
pt = stu;
```

向函数传递结构体的几个办法：

1.用结构体的单个成员作为函数的参数，向函数传递结构体的单个成员；传单个的参数

```
#include <stdio.h>

//建立结构
typedef struct lib
{
int data;
}LIB;
//求和的函数
int  fun(int c)
{
int d = 8;
return c+d;
}
int main()
{
    LIB b;
    int sum;
    b.data = 7;
    printf("修改之前的%d\n",b.data);//数据为7
    sum = fun(b.data);//调用作为单个参数的过程
    printf("计算之后的结果%d\n",sum);//数据为15
        return 0;
}
```

2.用结构体变量做函数参数，向函数传递结构体的完整结构。传递结构体变量

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

//建立结构
```

```
typedef struct lib
{
int data;
}LIB;
void fun(LIB p)
{
p.data = 8;
printf("修改之中的%d\n",p.data);
}LIB;
int main()
{
LIB b;
b.data = 7;
printf("修改之前的%d\n",b.data);//
fun(b);//调用整个结构体的过程
printf("修改之后的%d\n",b.data);
system("pause");
return 0;
}
```

3.用**结构体指针或者结构体数组作函数参数**，向函数传递结构体的地址。**传地址**

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

//建立结构
typedef struct date
{
int year;
int month;
int day;
}Date;

void fun(Date *p)
{
```

```
        p->year = 1995;
    p->month = 11;
    p->day = 18;
指向结构体的指针变量名->成员名
str->name  = wang;
    }
int main()
{
        Date b;
    b.year = 1999;
    b.month = 3;
    b.day = 10;
        printf("修改之前的生日%d-0%d-%d\n",b.year,b.month,b.day);
        fun(&b);//传递的时候需要传递一个地址的值
        printf("修改之后的生日%d-%d-%d\n",b.year,b.month,b.day);
        system("pause");
        return 0;
}
```

在传递结构体变量的时候，传递的变量的形式和传递普通的参数也没有区别。
只是传递当前值得一个副本，而不是传递地址过去。所以不会修改主函数中的值
使用的方法如下