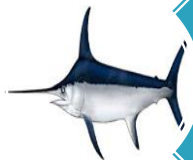


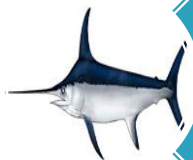
第八章 指针

蒋玉茹

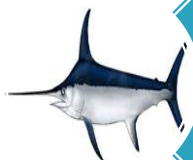
一、指针在程序中的用途



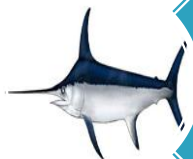
有效的表示复杂的数据结构



能动态分配内存



方便的使用字符串



直接处理内存地址

换房子中的东西



A: 200台笔记本

B: 5000部土豪



换钥匙

房子

- 存放数据值的内存空间

钥匙

- 内存空间的地址

指针

- 内存空间的地址

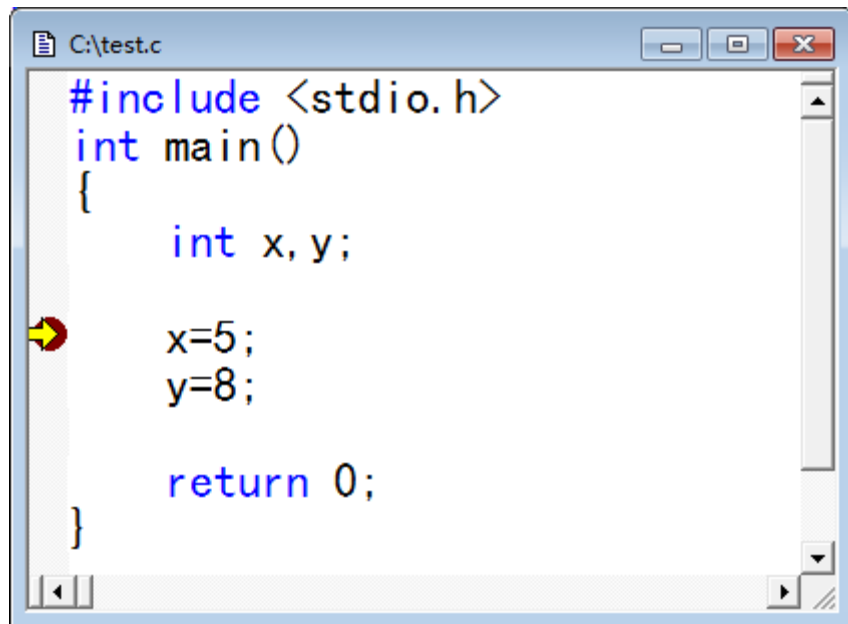
观察下面程序中变量的值和地址

```
#include <stdio.h>
int main()
{
    int x,y;

    x=5;
    y=8;

    return 0;
}
```

赋值之前



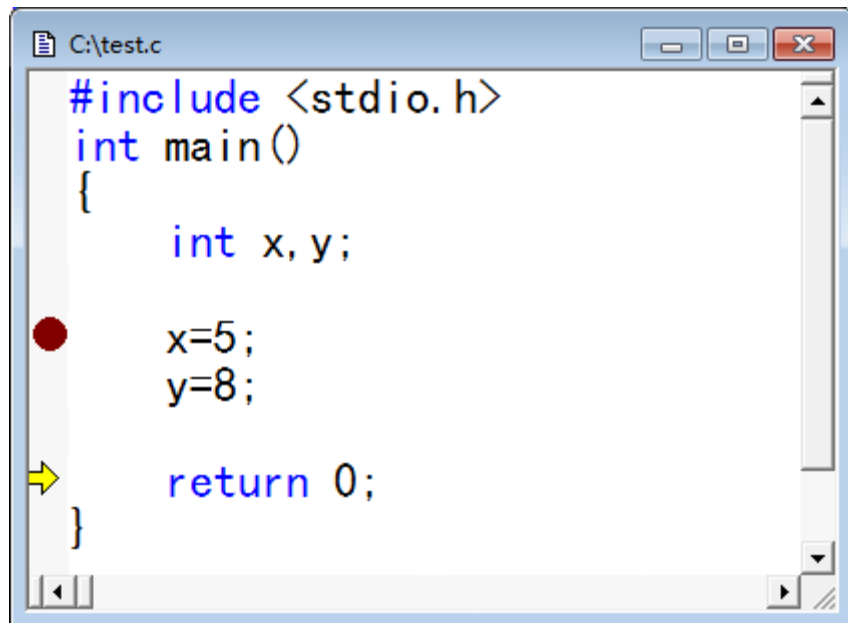
```
#include <stdio.h>
int main()
{
    int x, y;

    x=5;
    y=8;

    return 0;
}
```

名称	值
⊕ &x	0x0012ff44
⊕ &y	0x0012ff40
x	-858993460
y	-858993460

赋值之后



```
#include <stdio.h>
int main()
{
    int x, y;

    x=5;
    y=8;

    return 0;
}
```

名称	值
田 &x	0x0012ff44
田 &y	0x0012ff40
x	5
y	8

0x0012ff40

8

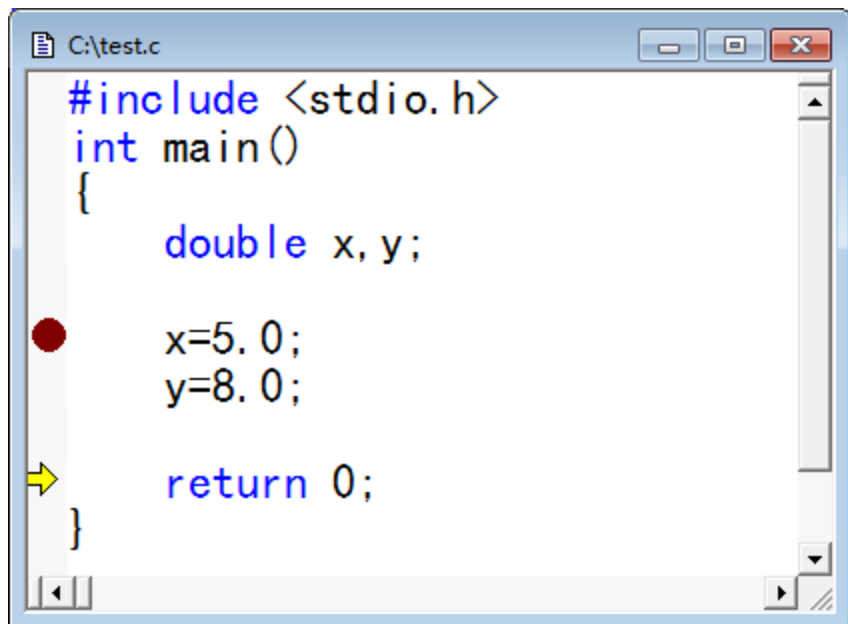
y

0x0012ff44

5

x

换个double型变量试试



```
#include <stdio.h>
int main()
{
    double x, y;

    x=5.0;
    y=8.0;

    return 0;
}
```

名称	值
⊕ &x	0x0012ff40
⊕ &y	0x0012ff38
x	5.0000000000000000
y	8.0000000000000000

0x0012ff38

8.0

y

0x0012ff40

5.0

x

0x0012ff40

8

y

0x0012ff44

5

x

0x0012ff38

8.0

y

0x0012ff40

5.0

x

怎样操纵钥匙呢？
怎样存取地址呢？

指针变量

- ▶ 存储地址的变量
- ▶ 定义：基类型 *指针变量名；
- ▶ 例：int *p; // p为指向整型变量的指针变量

怎样将钥匙和房子联系在一起

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

```
int main()
```

```
{
```

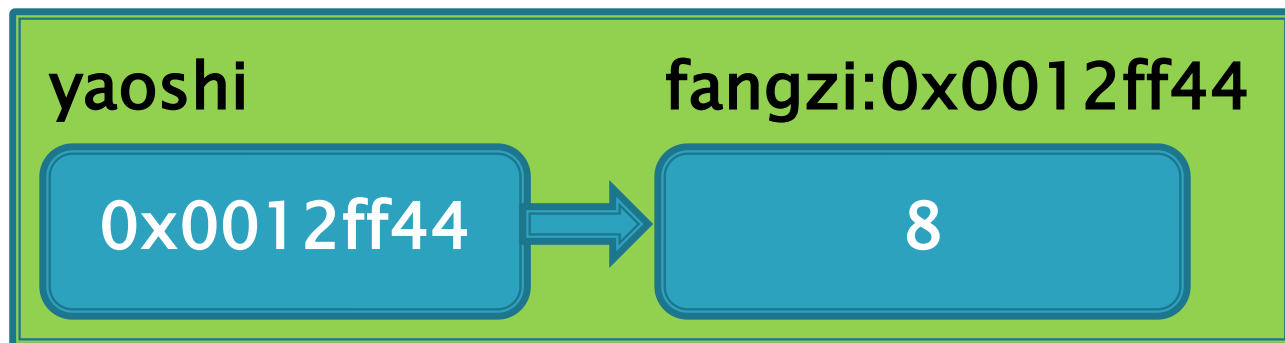
```
    int fangzi=8; //定义房子
```

```
    int *yaoshi;  //定义钥匙
```

```
    yaoshi=&fangzi; //钥匙与房子关联
```

```
    return 0;
```

```
}
```



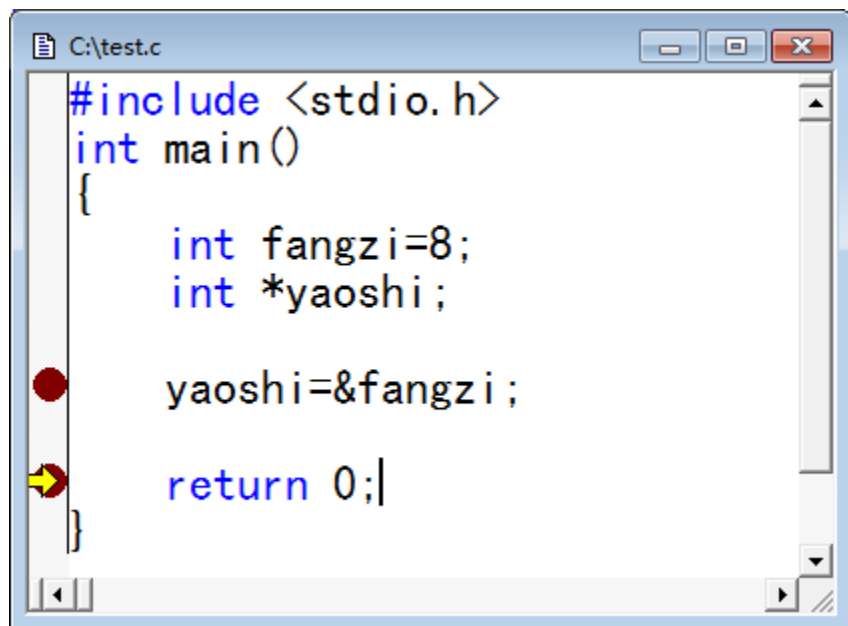
怎样通过钥匙取得房子中的东西

```
#include <stdio.h>
int main()
{
    int fangzi=8;    //定义房子
    int *yaoshi;     //定义钥匙

    yaoshi=&fangzi; //钥匙与房子关联

    printf("%d",*yaoshi);
    return 0;
}
```


运行程序并观察



```
C:\test.c
#include <stdio.h>
int main()
{
    int fangzi=8;
    int *yaoshi;

    yaoshi=&fangzi;

    return 0;|
}
```

名称	值
fangzi	8
▣ yaoshi	0x0012ff44
▣ &fangzi	0x0012ff44
*yaoshi	8

总结：访问变量的两种方式

直接访问方式

操纵变量的名字

间接访问方式

操纵指向变量的指针变量

基类型的作用

指针变量的定义：**基类型** ***指针变量名；**

指针变量
的基类型



指针变量
指向的变
量的类型

使用指针时的两个神器

*

定义指针变量

取指针变量指向
的变量的内容

&

取变量的地址

练习：读程序

```
#include <stdio.h>

int main()
{   int    a,b;
    int *p1,*p2;
    a=100;b=10;
    p1=&a;p2=&b;
    printf("%d, %d\n",a,b);
    printf("%d, %d\n",*p1,*p2);
    return 0;
}
```

练习：读程序

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

```
int main()
```

```
{   int    *p1, *p2, *p, a, b;
```

```
    scanf("%d%d",&a,&b);
```

```
    p1=&a;  p2=&b;
```

```
    if(a<b) {   p=p1; p1=p2;  p2=p; }
```

```
    printf("%d%d",a,b);
```

```
    printf("ma=%d,mi=%d",  *p1,  *p2 );
```

```
    return 0;
```

```
}
```

你能画出程序中指针处理过程的示意图吗？

练习：读程序

```
#include <stdio.h>
int main()
{ void swap(int x,int y);
  int a,b;
  int*po1,*po2;
  scanf("%d,%d",&a,&b);
  po1=&a;
  po2=&b;
  if(a<b)
      swap(a,b);
  printf("\n%d,%d\n",a,b);
  printf("\n%d,%d\n",*po1,*po2);
  return 0;
}
```

```
void swap(int x,int y)
{
  int temp;
  temp=x;
  x=y;
  y=temp;
}
```

练习：读程序

```
#include <stdio.h>
int main()
{ void swap(int *p1,int *p2);
  int a,b;
  int*po1,*po2;
  scanf("%d,%d",&a,&b);
  po1=&a;
  po2=&b;
  if(a<b)
    swap(po1,po2);
  printf("\n%d,%d\n",a,b);
  printf("\n%d,%d\n",*po1,*po2);
  return 0;
}
```

```
void swap(int *p1,int *p2)
{
  int temp;
  temp=*p1;
  *p1=*p2;
  *p2=temp;
}
```


指针第二讲（第2课时）

复习指针的基本概念

```
#include <stdio.h>
int main()
{
    int a,b;
    int *pa,*pb;
    pa=&a;
    pb=&b;
    scanf("%d%d",pa,pb);
    //此处插入代码，实现输入数据的从小到大输出

    printf("%d,%d",*pa,*pb);
    return 0;
}
```

练习：读程序

```
#include <stdio.h>
int main()
{ void swap(int *p1,int *p2);
  int a,b;
  int*po1,*po2;
  scanf("%d,%d",&a,&b);
  po1=&a;
  po2=&b;
  if(a<b)
    swap(po1,po2);
  printf("\n%d,%d\n",a,b);
  printf("\n%d,%d\n",*po1,*po2);
  return 0;
}
```

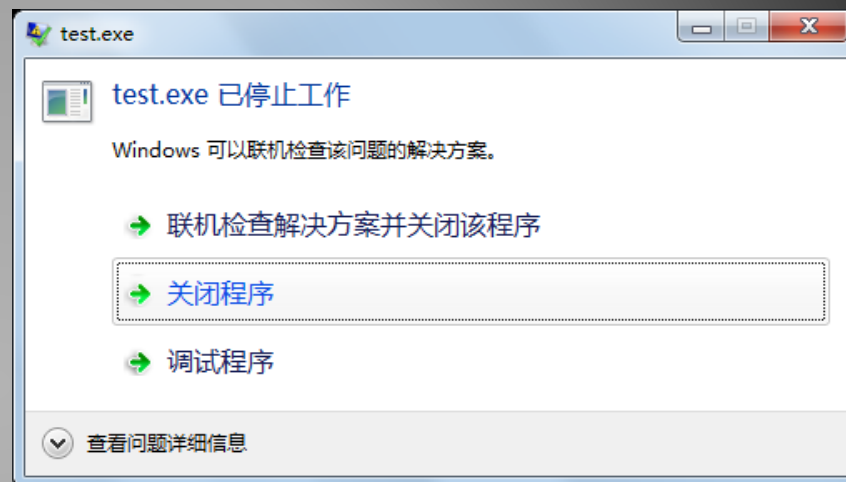
```
void swap(int *p1,int *p2)
{
  int *p;
  p=p1;
  p1=p2;
  p2=p;
}
```

观察下面的程序

```
#include <stdio.h>
int main()
{
    int *p;

    printf("%d",*p);

    return 0;
}
```



没有房子的钥匙

练习：读程序

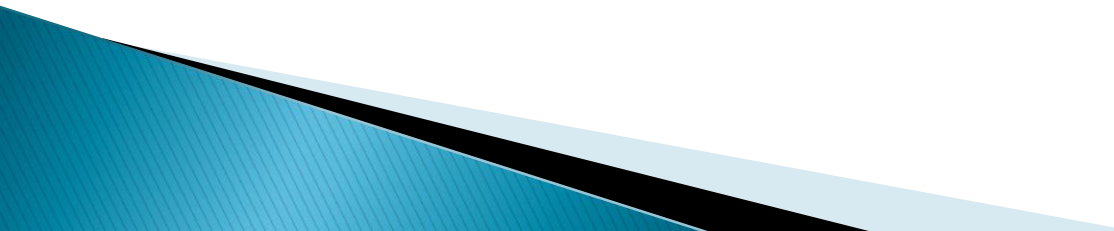
```
#include <stdio.h>
int main()
{ void swap(int *p1,int *p2);
  int a,b;
  int*po1,*po2;
  scanf("%d,%d",&a,&b);
  po1=&a;
  po2=&b;
  if(a<b)
    swap(po1,po2);
  printf("\n%d,%d\n",a,b);
  return 0;
}
```

```
void swap(int *p1,int *p2)
{
  int *temp;
  *temp=*p1;
  *p1=*p2;
  *p2=*temp;
}
```

练习：编程

- ▶ 学生成绩管理，5门课10名学生，编写一个自定义函数，该函数的作用是找到所有成绩中最高成绩所在的行和列。主函数的功能是学生成绩的录入，及最高成绩所在行、列的输出。
- ▶ `void findHighestScore(float score[][5],int *hang,int *lie)`

练习：编程

- ▶ 编写如下函数，求出并设置y年m月d日的前一天或后一天的日期（能正确判断闰年）。
 - ▶ `void yesterday(int *y,int *m,int *d)`
 - ▶ `void tomorrow(int *y,int *m,int *d)`
- 

练习：编程

- ▶ 编写如下函数，将三个int型整数按升序排列
- ▶ `void sort3(int *n1,int *n2,int *n3)`

指针第三讲（第4课时）

数组与指针

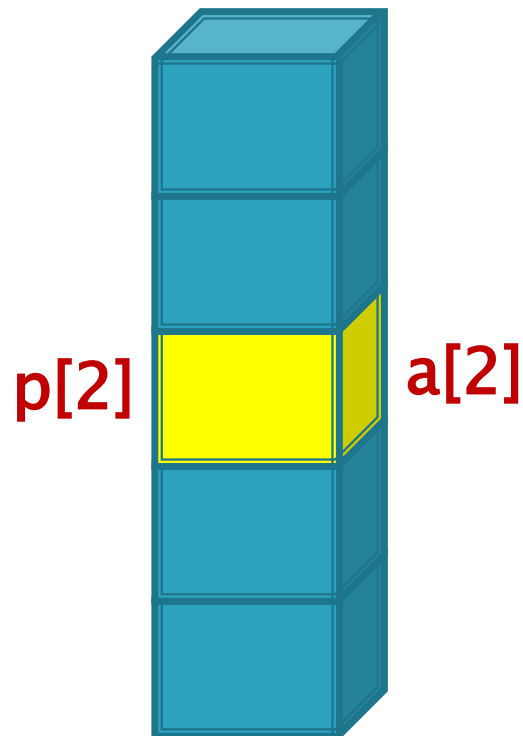
数组名是什么？

- ▶ `int a[10];`
- ▶ 数组名：a
- ▶ 数组元素类型：int
- ▶ 数组长度：10
- ▶ a是地址
- ▶ a是数组第一个元素的地址，即数组的首地址



数组和指针

1. `int a[10];`
2. `int *p;`
3. `p=a;`
4. `p=&a[0];`
5. `p[2]`和`a[2]`什么关系?



利用指向数组的指针操纵数组元素

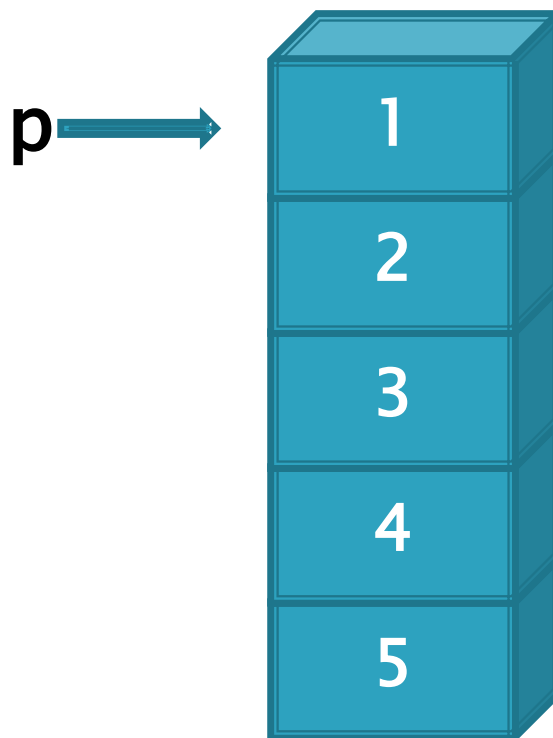
```
#include <stdio.h>
int main()
{
    int a[10]={1,2,3,4,5};
    int *p;
    int i;
    p=a;
    for(i=0;i<5;i++)
    {
        printf("%d ",p[i]);
    }
    return 0;
}
```

利用指向数组的指针操纵数组元素

```
#include <stdio.h>
int main()
{
    int a[10]={1,2,3,4,5};
    int *p;
    int i;
    p=a;
    for(i=0;i<5;i++)
    {
        printf("%d ",*(p+i));
    }
    return 0;
}
```

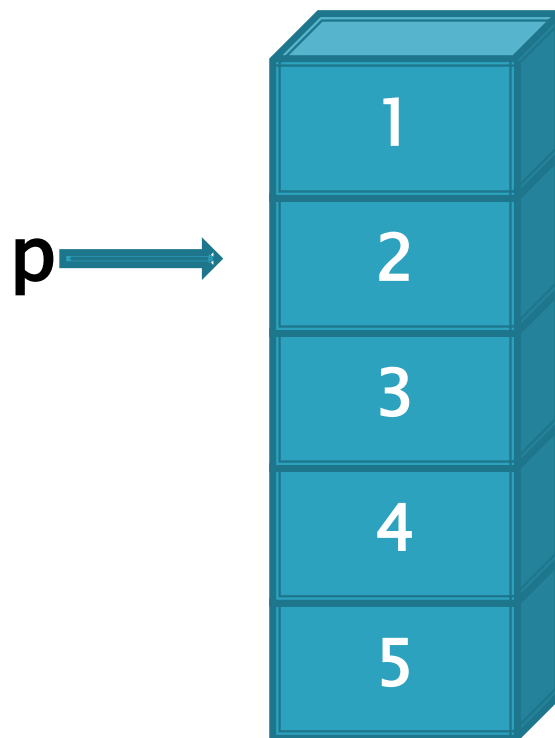
指针可以移动

```
#include <stdio.h>
int main()
{
    int a[10]={1,2,3,4,5};
    int *p;
    int i;
    p=a;
    for(i=0;i<5;i++)
    {
        printf("%d ",*p);
        p++;
    }
    return 0;
}
```



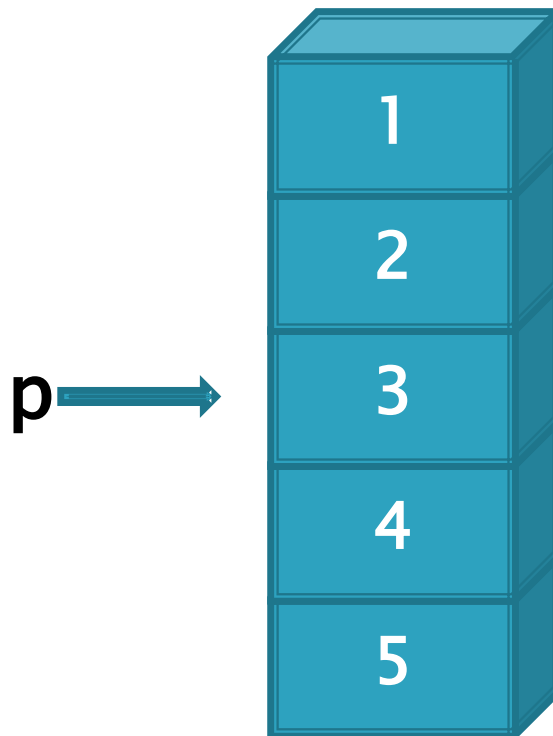
指针可以移动

```
#include <stdio.h>
int main()
{
    int a[10]={1,2,3,4,5};
    int *p;
    int i;
    p=a;
    for(i=0;i<5;i++)
    {
        printf("%d ",*p);
        p++;
    }
    return 0;
}
```



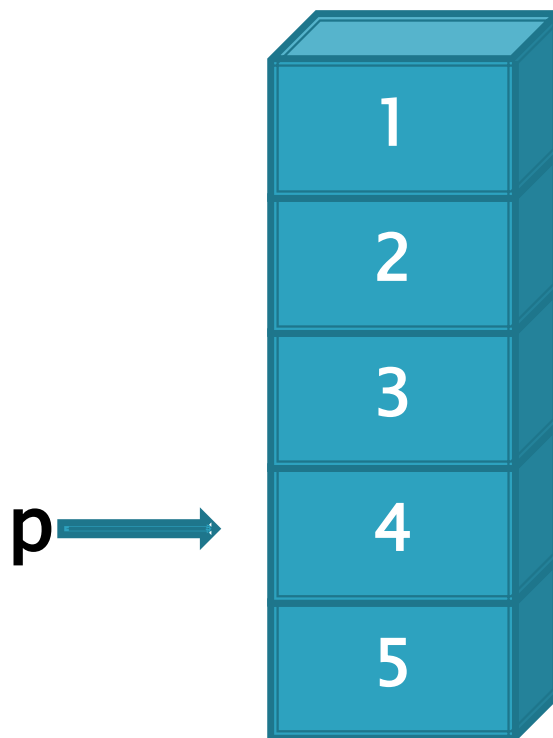
指针可以移动

```
#include <stdio.h>
int main()
{
    int a[10]={1,2,3,4,5};
    int *p;
    int i;
    p=a;
    for(i=0;i<5;i++)
    {
        printf("%d ",*p);
        p++;
    }
    return 0;
}
```



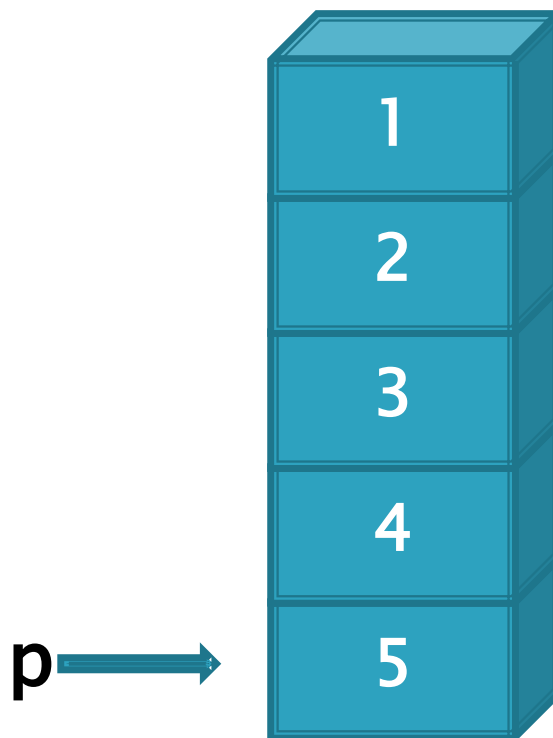
指针可以移动

```
#include <stdio.h>
int main()
{
    int a[10]={1,2,3,4,5};
    int *p;
    int i;
    p=a;
    for(i=0;i<5;i++)
    {
        printf("%d ",*p);
        p++;
    }
    return 0;
}
```



指针可以移动

```
#include <stdio.h>
int main()
{
    int a[10]={1,2,3,4,5};
    int *p;
    int i;
    p=a;
    for(i=0;i<5;i++)
    {
        printf("%d ",*p);
        p++;
    }
    return 0;
}
```

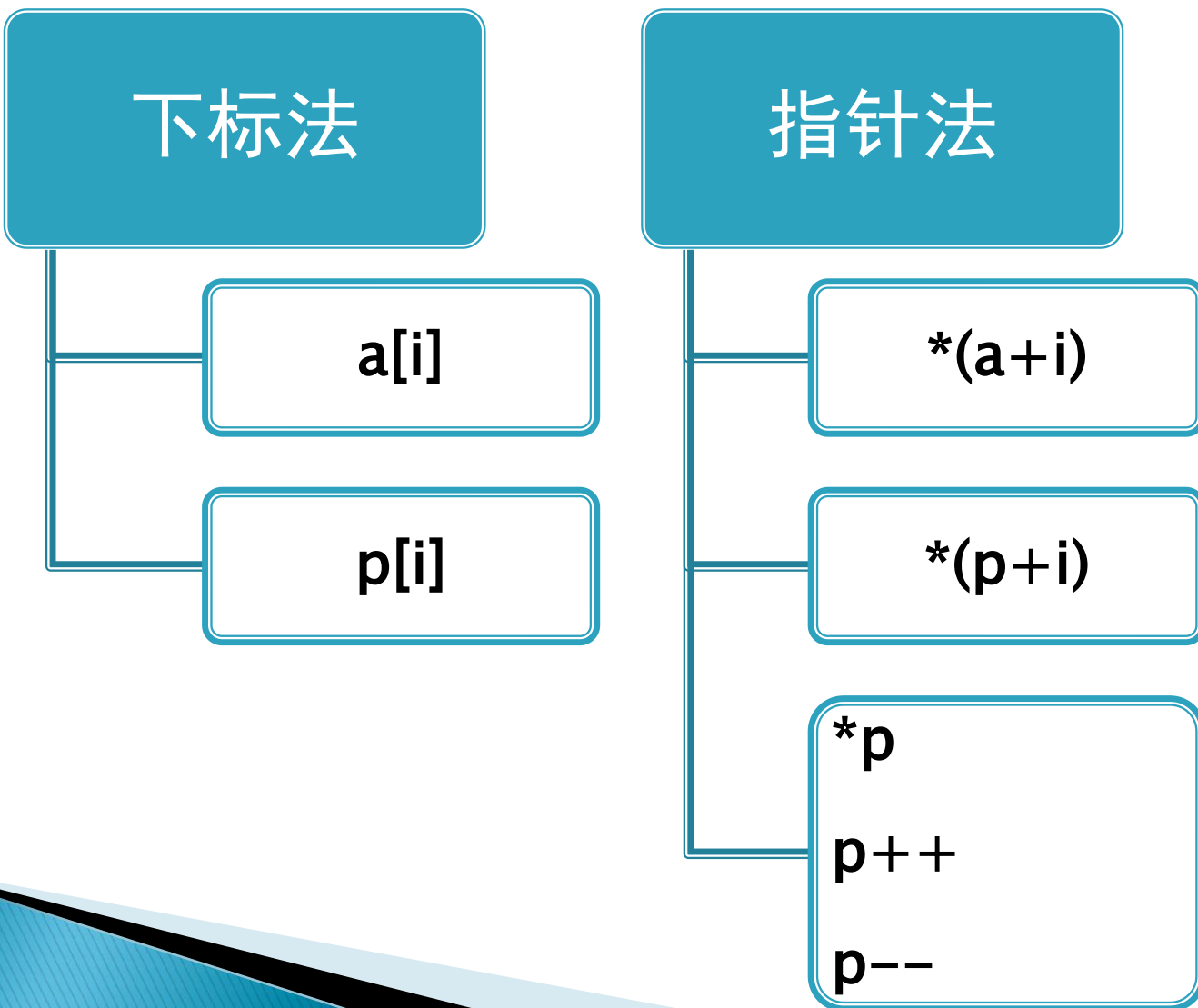


练习：读程序

```
#include <stdio.h>
int main()
{
    int a[10]={1,2,3,4,5};
    int *p;
    int i;
    p=a;
    for(i=0;i<5;i++)
    {
        printf("%d ",*p);
        p++;
    }
}
```

```
for(i=0;i<5;i++)
{
    printf("%d ",*p);
    p--;
}
return 0;
}
```

小结：操纵数组元素的方式



练习：编程

编写自定义函数：

```
void square(int  
*vc,int no,int val)
```

其功能是将vc所指向的数组中前no个数组元素的值改为val。

编写main函数调用square测试其功能

```
#include <stdio.h>
int main()
{
    void square(int *vc,int no,int val);
    int array[10]={1,2,3,4,5,6};
    int i,m_no=3,m_val=3;

    square(array,m_no,m_val);

    for(i=0;i<m_no;i++)
    {
        printf("%2d",array[i]);
    }

    return 0;
}

void square(int *vc,int no,int val)
{
}
```

函数间传递数组的方式

实参和形参都为数组名

调用函数

```
int ma[10];  
.....  
fun(ma, .....);
```

被调用函数

```
void fun(int a[], .....)  
{  
  
}  
}
```


实参为数组名，形参为指针变量

调用函数

```
int ma[10];  
.....  
fun(ma, .....);
```

被调用函数

```
void fun(int *p, .....)  
{  
  
}  
}
```

实参为指向数组的指针变量，形参为指针变量

调用函数

```
int ma[10],*mp;  
mp=ma;  
.....  
fun(mp, .....);
```

被调用函数

```
void fun(int *p, .....)  
{  
  
}  
}
```

实参为指向数组的指针变量，形参为数组名

调用函数

```
int ma[10],*mp;  
mp=ma;  
.....  
fun(mp, .....);
```

被调用函数

```
void fun(int a[], .....)  
{  
  
}  
}
```

字符数组与指针

练习：编程

- ▶ 自定义一个函数，其功能是将一个字符串变成它的逆序字符串
- ▶ 编写main函数测试其功能

```
#include <stdio.h>
int main()
{
    void inverse(char *vc,int length);
    char array[10]="hello";
    int i;

    inverse(array,5);

    for(i=0;i<5;i++)
    {
        printf("%c",array[i]);
    }
    return 0;
}
void inverse(char *vc,int length)
{
}
```

怎样存储10个人的名字？

```
#include <stdio.h>
int main()
{
    char a[10][20];
    char *pa;
    int i=0;
    pa=&a[0][0];
    for(i=0;i<10;i++)
    {
        scanf("%s",pa);
        pa=&a[i][0];
    }
}
```

```
pa=&a[0][0];
for(i=0;i<10;i++)
{
    printf("%s\n",pa);
    pa=&a[i][0];
}
return 0;
}
```

指向数组的指针

- ▶ `char (*pa)[20];` // 指向长度为20的字符数组的指针
- ▶ `pa++;` // 指针向前移动20个字符
- ▶ `pa--;` // 指针向后移动20个字符
- ▶ 与二维数组配合使用
 - `char a[10][20];`
 - `char (*pa)[20];`
 - `pa=a;`


```
#include <stdio.h>
int main()
{
    char a[10][20];
    char (*pa)[20];
    int i=0;
    pa=a;
    for(i=0;i<10;i++)
    {
        scanf("%s",pa);
        pa++;
    }
}
```

```
pa=a;
for(i=0;i<10;i++)
{
    printf("%s\n",pa);
    pa++;
}
return 0;
}
```

指针数组

- ▶ 一个数组，若其元素均为指针类型数据，称为指针数组。
- ▶ `int *p[10];`
- ▶ `float *p[10];`

练习：形成下面的态势

字符串

Follow me
BASIC
Great Wall
FORTRAN
Computer design

(a)

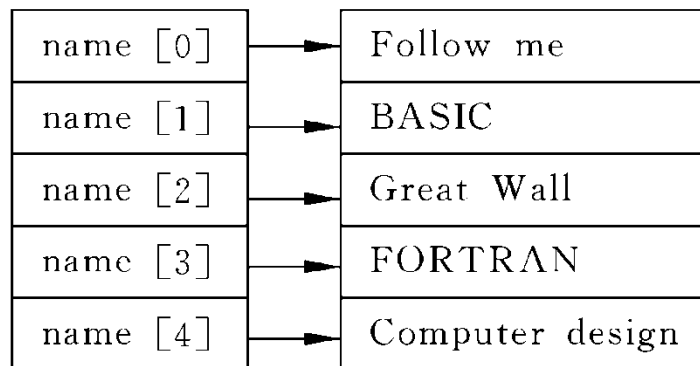
F	o	l	l	o	w		m	e	\0						
B	A	S	I	C	\0										
G	r	e	a	t		W	a	l	l	\0					
F	O	R	T	R	A	N	\0								
C	o	m	p	u	t	e	r		d	e	s	i	g	n	\0

(b)

name

指针数组

字符串



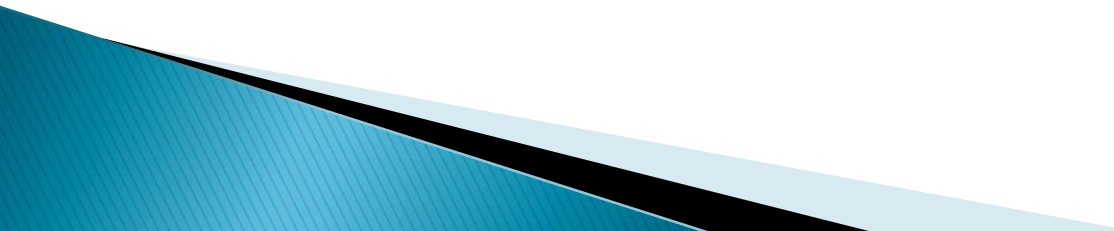
(c)

练习：编程

- ▶ 对上面的字符串排序
- ▶ 换钥匙还是换仓库？

字符串常量 与指向字符指针

```
#include <stdio.h>
#include <string.h>
int main()
{
    char *str = "china";
    puts(str);
    return 0;
}
```



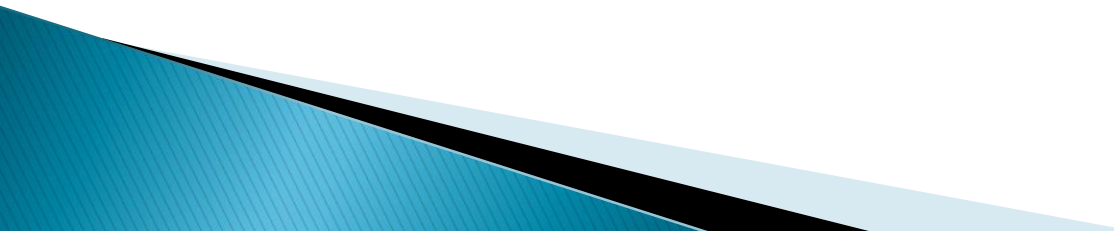
字符串常量与指针数组

练习：添加代码，使得数组中的字符串按照升序顺序依次输出

```
#include <stdio.h>
#include <string.h>
int main()
{
    char *str[5]={"england",
                  "china",  "japan",
                  "american","korean"};
    int i=0;
    // 此处添加代码
```

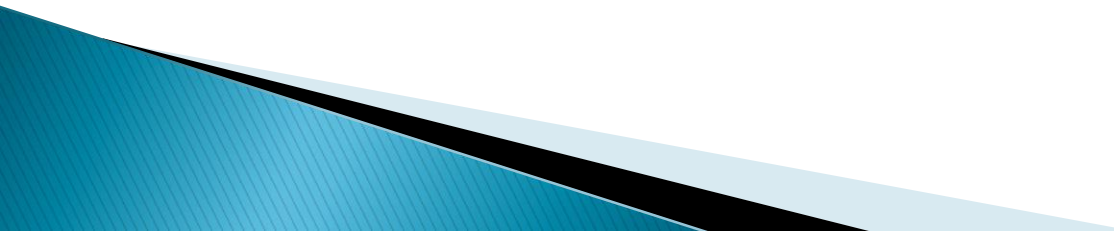
```
        for(i=0;i<5;i++)
            puts(str[i]);
        return 0;
    }
```


词典排序

- ▶ 初始化词典内容
 - ▶ 输出词典的原始内容
 - ▶ 对词典进行升序或者降序排列
 - 排序算法选择用冒泡或者选择排序
 - 排序中交换内容？或者指针？
 - ▶ 输出排序后词典的内容
- 

动态内存分配

头文件stdlib.h中的几个库函数

- ▶ `void * malloc(unsigned int size);`
 - ▶ `void * calloc(unsigned n, unsigned size);`
 - ▶ `void free(void *p);`
 - ▶ `void * realloc(void *p, unsigned int size);`
- 

主要的库函数

- ▶ `void * malloc(unsigned int size)`
- ▶ 功能：在内存的动态存储区中分配一个长度为size的连续空间
- ▶ 参数：size，无符号整数，申请空间的大小（字节）
- ▶ 返回值：void *，分配空间的第一个字节的地址；分配不成功则返回空指针（NULL）
- ▶ `int * p;`
- ▶ `p=(int *)malloc(sizeof(int));`

主要的库函数

- ▶ `void * calloc (unsigned n , unsigned size)`
- ▶ 功能：在内存的动态存储区中分配n个长度为size的连续空间，可以为二维数组开辟动态存储空间
- ▶ 参数：n为数组元素个数，每个元素占用size个字节
- ▶ 返回值：void *，分配空间的第一个字节的地址；分配不成功则返回空指针（NULL）
- ▶ `int * p;`
- ▶ `p=(int *)calloc(5,sizeof(int));`

主要的库函数

- ▶ `void free(void *p)`
- ▶ 功能：释放指针变量p所指向的动态空间，使这部分空间能重新被其他变量使用。
- ▶ 参数：p为用malloc或者calloc分配的空间的起始地址
- ▶ 返回值：无

- ▶ `int * p;`
- ▶ `p=(int *)calloc(5,sizeof(int));`
- ▶ `.....`
- ▶ `free(p);`

主要的库函数

- ▶ `void *realloc(void *p , unsigned int size)`
- ▶ 功能：如果已经通过malloc或者calloc获得了动态空间，p为其首地址，而且已经向这个空间中存储了内容，可是发现这个空间不够大，此时可以通过realloc函数重新分配空间的大小。Realloc函数将p所指向的动态空间的大小改变为size。P的值不变。
- ▶ 参数：p为用malloc或者calloc分配的空间的起始地址
- ▶ 返回值：如果分配不成功，则返回NULL

```
int * p;  
p=(int *)malloc(1024);  
.....  
realloc(2048);  
.....  
free(p);
```

练习

- ▶ 编写一个程序实现10个句子的输入和输出
- ▶ 句子的长度不一，短的有3个字母，长的多大500个字母。因此需要用malloc动态分配内存
- ▶ 要求存放句子的内存空间初始长度为10个字符；如果不够，以原来空间的2倍重新分配内存空间


```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define SNTLEN 10//句子长度
#define SNTSIZE 2//句子个数
int main()
{
    char *snt[SNTSIZE];
    int i,j;

    for(i=0;i<SNTSIZE;i++)
    {
        snt[i]=(char *)malloc(SNTLEN*sizeof(char));
        *(snt[i]+SNTLEN*sizeof(char)-1)='\0';
        j=-1;
        //此处代码见下页
        *(snt[i]+j+1)='\0';
    }
    //后续代码见下下页
```

读取一个句子

```
do{
    j++;
    if(j==strlen(snt[i]))
    {
        snt[i]=(char *)realloc(snt[i],2*strlen(snt[i])*sizeof(char));
        *(snt[i]+2*strlen(snt[i])*sizeof(char)-1)='\0';
    }
    *(snt[i]+j) = getchar();
}while((*snt[i]+j))!='\n');
```

输出所有句子内容

```
for(i=0;i<SNTSIZE;i++)  
{  
    puts(snt[i]);  
}
```

释放申请过的动态内存

```
for(i=0;i<SNTSIZE;i++)  
{  
    free(snt[i]);  
}  
return 0;  
}
```

异常处理

```
snt[i]=(char *)malloc(SNTLEN*sizeof(char));  
if(snt[i]==NULL)  
{  
    printf("内存不足， 程序退出！");  
    return 0;  
}
```

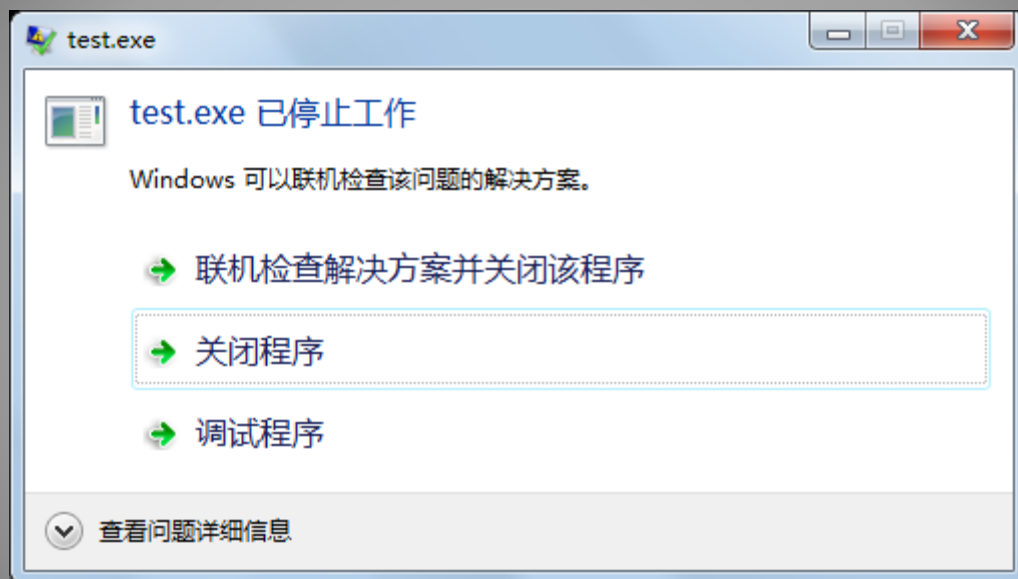
- ▶ 凡是在动态内存申请之后都需要做此异常处理

练习：编程【这个实例尚未构造好】

- ▶ 人为制造两个大字符串
- ▶ 交换 n 次

```
#include <stdio.h>
#include <string.h>
#define N 200
int main()
{
    char a[N],b[N],c[N];
    int i;
    for(i=0;i<N;i++)
    {
        a[i]='a';
        b[i]='b';
    }
}
```

```
for(i=0;i<N;i++)
{
    strcpy(c,a);
    strcpy(a,b);
    strcpy(b,c);
}
return 0;
}
```




```
#include <stdio.h>
#include <string.h>
#define N 200
int main()
{
    char a[N],b[N],c[N];
    int i;
    for(i=0;i<N;i++)
    {
        a[i]='a';
        b[i]='b';
    }
    a[N-1]='\0';
    b[N-1]='\0';
```

```
    for(i=0;i<N;i++)
    {
        strcpy(c,a);
        strcpy(a,b);
        strcpy(b,c);
    }
    return 0;
}
```

```
#include <stdio.h>
#include <string.h>
#define N 200000
int main()
{
    char a[N],b[N];
    char *pa,*pb,*pc;
    pa=a;pb=b;
    int i;
    for(i=0;i<N;i++)
    {
        a[i]='a';
        b[i]='b';
    }
}
```

```
for(i=0;i<N;i++)
{
    pc=pa;
    pa=pb;
    pb=pc;
}
return 0;
}
```