C语言程序设计

计算机科学与技术学院

© 东华大学

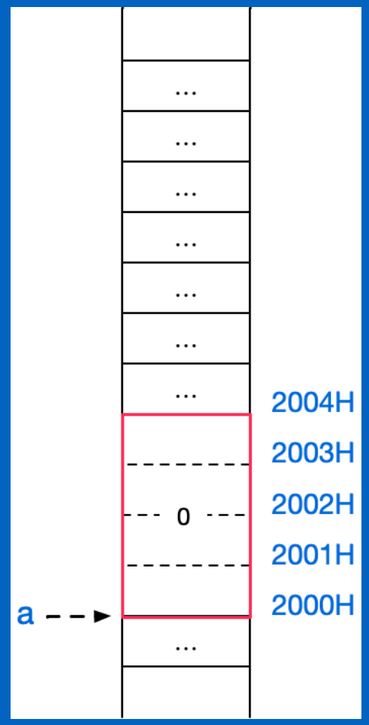
指针与数组

© 东华大学

回顾:变量与地址

- 地址
  - 内存中每个存储单元的线性序号
- 变量
  - 编译或函数调用时为变量分配内存空间
  - 变量名称：存储空间起始地址的别名

```
int a = 0;
```

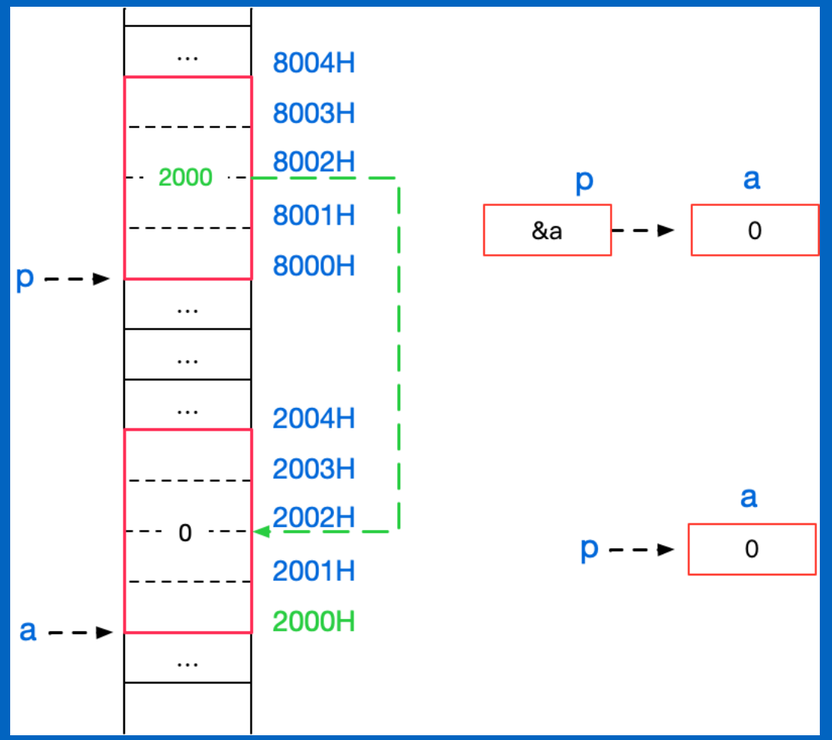


© 东华大学

地址与指针变量

- 地址
  - 内存中每个存储单元的线性序号
- 指针：变量在内存中存储的首地址称为该变量的指针
  - 精化？糟粕?
- 指针变量: *pointer*
  - 专门存放变量地址的变量
- 一般形式： <类型> *<指针名>
  - 类型：指针的目标变量的数据类型
  - * ：表示定义指针变量，不是运算符
  - 指针名：合法标识符

```
int a = 0;

int *p; //定义指针

p = &a;
```

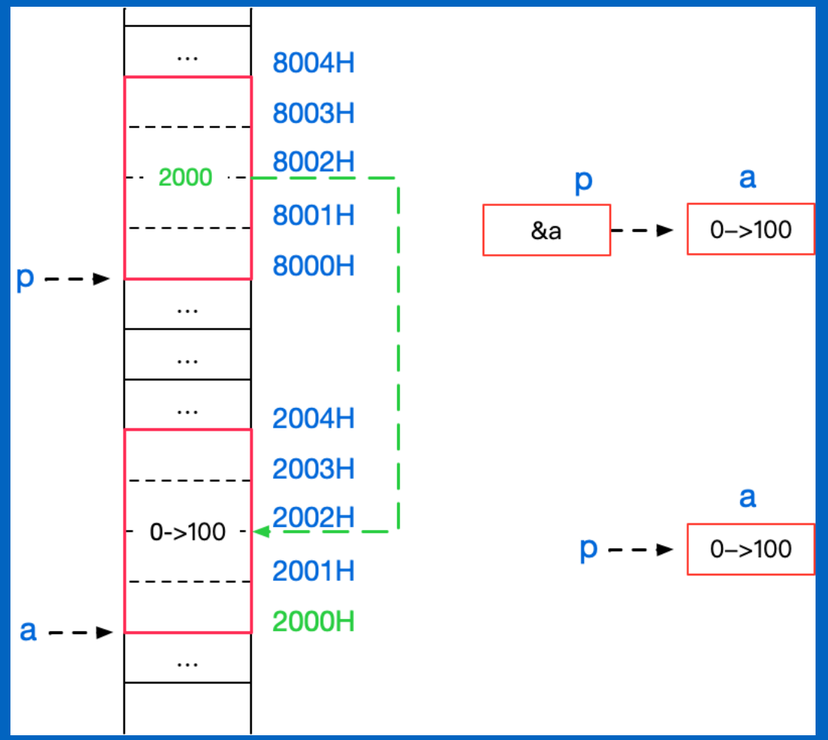


© 东华大学

地址与指针变量

- 指针变量定义：变量名前带 *
- 取址运算 (&)：取变量的地址
  - 优先级: 2
  - 结合性:自右向左
- 取值运算 (*)：访问指针所指向的变量
  - 优先级: 2
  - 结合性:自右向左
- 赋值运算 (=): 指向某个地址
- 以下两个 * 的作用不同：

```
int a = 0;

int *p; //定义指针,*是说明符

p = &a;

*p = 100;// a = 100; *是运算符
```



© 东华大学

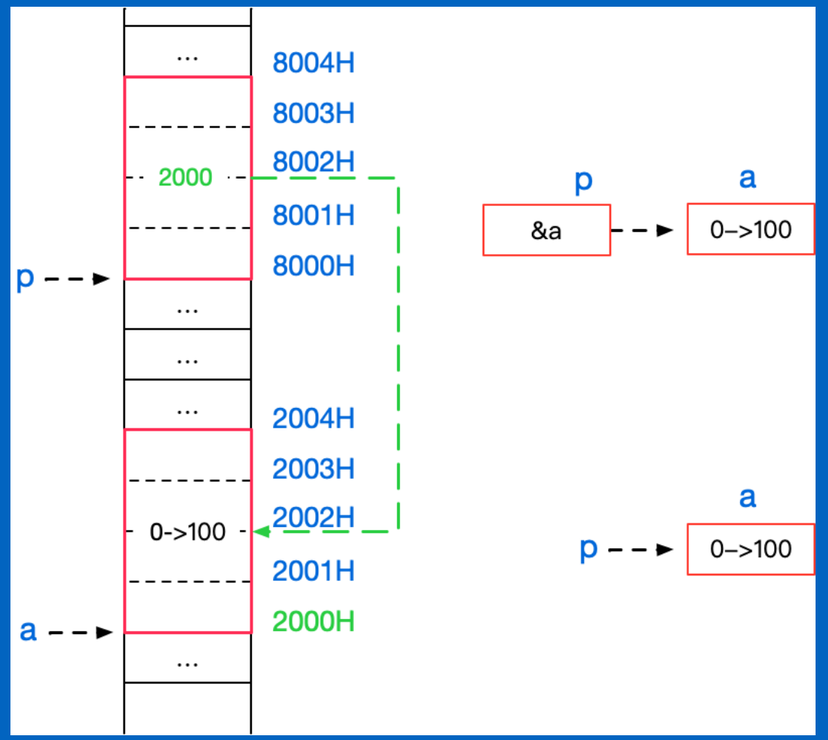
指针的运算

```
int a = 0, *p; //定义指针,*是说明符
p = &a;
*p = 100;// a = 100; *是运算符
```

则:

- p：指针变量，它的内容是地址量
- *p：指针的目标变量，它的内容是数据
- &p：指针变量占用内存的地址
- *与 & 是互逆的运算:

  a = *p = *(&a)



© 东华大学

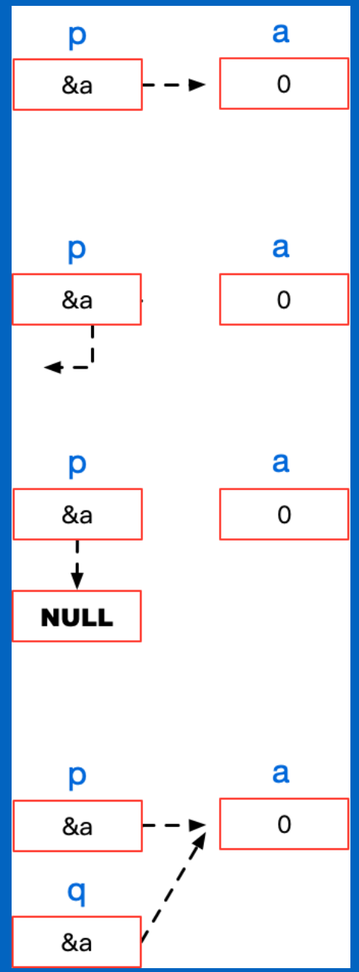
指针变量的初始化

- 一般形式：<类型> *<指针名>=<初值>；
- 指针只能存放地址值，在给指针赋值时必须使用变量的地址值表达式
  - int a = 0, *p = &a;
- 指针必须先指向某个变量，然后才能使用该指针
  - int a = 0, *p; a = *p; //逻辑错误，p未初始化
- 如果指针不指向任何变量，置该指针为NULL，NULL称为空指针

```
int a = 0, *p = NULL ;
.....                  //中间代码略（可能会对p赋值）
if (p != NULL)//当指针是一个有效指针时
    a = *p;//指针所指的值赋值给x
```

- 一个指针可以用相同类型的已知指针给它赋值

```
int a = 0, *p = &a, *q ;
float *f;
q = p;//q与p指向同一对象（q指向a）
f = p;//错误，f 与 p 不是指向相同类型的指针
```

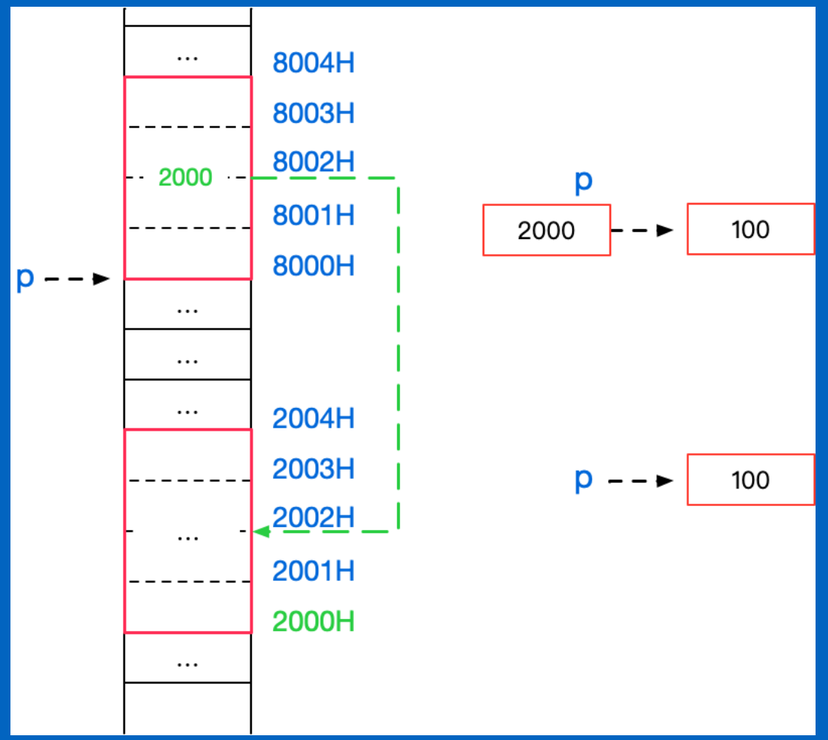


© 东华大学

指针变量的初始化

- 指针可以使用内存分配函数malloc()赋值
- 所指向的地址没有其他“变量名”对应

```
int *p;
p=(int *)malloc(sizeof(int));
if (p != NULL)//当指针是一个有效指针时
    *p = 100;//指针所指的值赋值给x
```

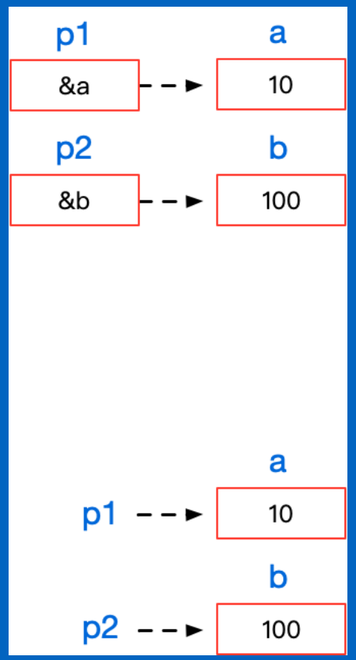


© 东华大学

变量的直接访问与间接访问

- 变量的直接访问：直接使用变量的名称
- 变量的间接访问：通过指针实现变量的间接访问

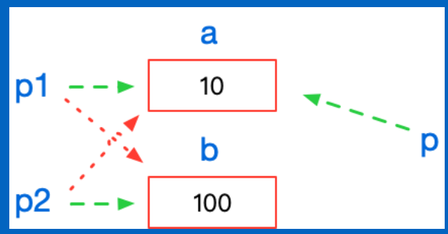
```c
#include <stdio.h>
int main (){
    int a, b;
    int *p1, *p2;

    a = 100;
    b = 10;

    p1 = &a;
    p2 = &b;

    //直接访问
    printf("%d,%d\n", a, b);
    //间接访问
    printf("%d,%d\n", *p1, *p2);

    return 0;
}
```



© 东华大学

输入两个数并使其从大到小输出

```c
#include <stdio.h>
int main (){
    int a, b;
    int *p1, *p2, *p;

    p1 = &a;
    p2 = &b;

    scanf("%d,%d", p1, p2);
    if(*p1 < *p2){
        p = p1;
        p1 = p2;
        p2 = p;
    }

    printf("a=%d,b=%d\n", a, b);
    printf("max=%d,min=%d\n", *p1, *p2);

    return 0;
}
```

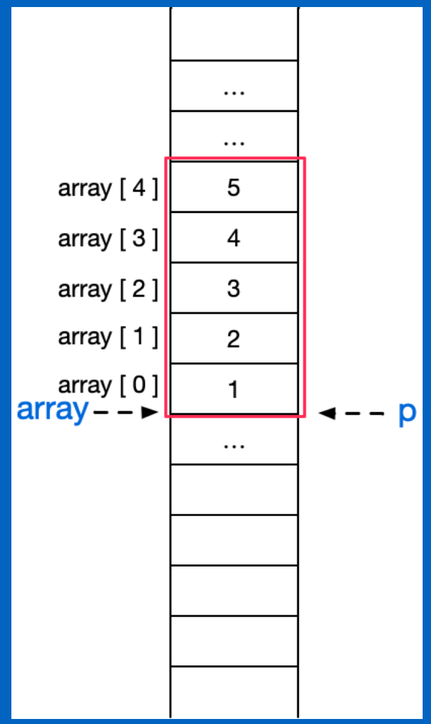


© 东华大学

数组与指针

- 指向数组的指针:数组的指针是指数组的起始地址
- 指向数组元素的指针:数组元素的指针是数组元素的地址

```
int array[10], *p; //定义指针,*是说明符

p = &array[0]; // p = array;
```

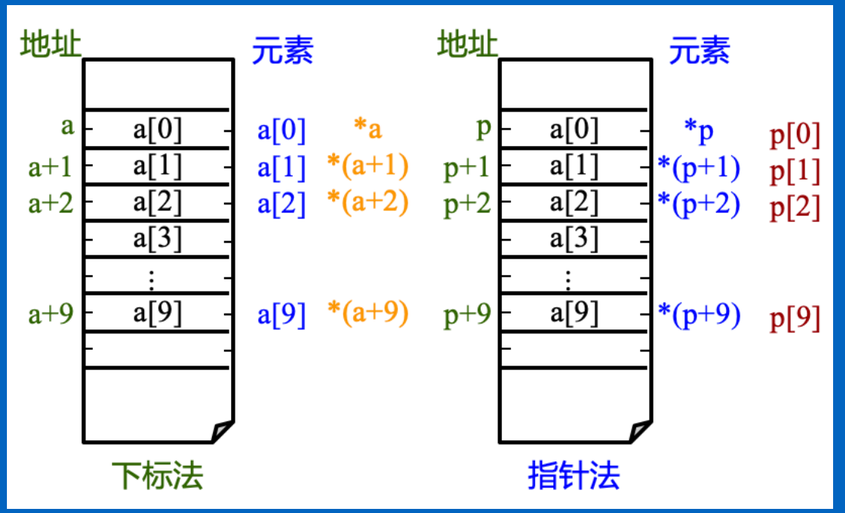


© 东华大学

一维数组元素的指针表示

```
int    a[10];
int    *p = a;
```

- 数组元素的表示方法：
  - a[i]
  - p[i]
  - *(p+i)
  - *(a+i)

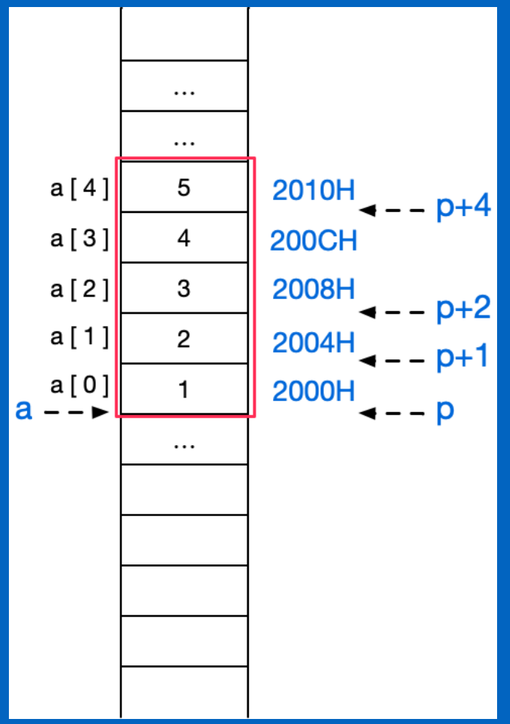


© 东华大学

指针的算术运算

- 自增、自减运算
  - p++, p--: 使p从当前所指向的数据移到指向后一个数据或者前一个数据
  - 指针变量自增1或者自减1并不表示其存放的地址值加1或者减1
  - 地址值的增加值或减少值取决于指针变量所指对象占用的字节数

```
//若a的地址为2000
int  a[10], p = a;

p++;  // p 指向 a[1]
            // 即p指向 2000+sizeof(int)=2004

++p;  // p 指向 a[2]
p--;  // p 指向 a[1]
--p;  // p 指向 a[0]
```

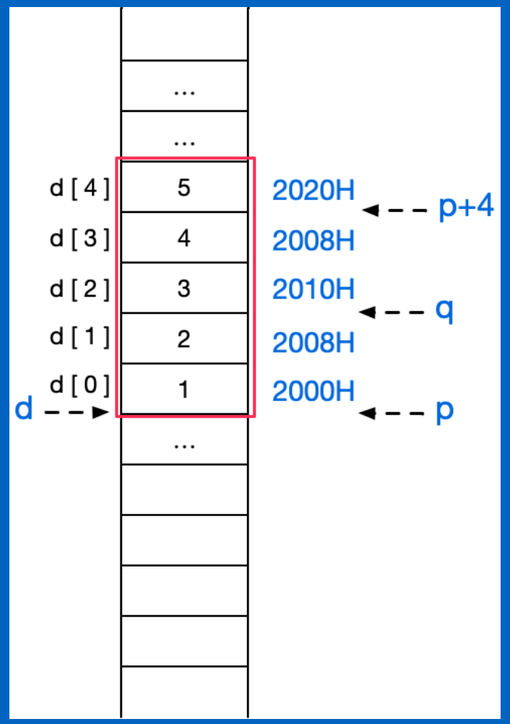


© 东华大学

指针的运算

- 增加 n 、减少 n
  - p+n, p-n: 指向当前所指的那个变量的后面（或前面）第n个数据
- 指针变量相减
  - 两个地址差之间能存放的这种类型的数据的个数

```
//若f的地址为2000
double d[10], *p = d, *q = &d[2];

p = p + 4;
//p 指向2000+4*sizeof(double)
//    =2000+4*8=2020H(十六进制)

printf("%d", p - q);// p - q = 2;
```



© 东华大学

输出数组中的全部元素

- 下标法

```
#include <stdio.h>
int main (){
    int a[10];
    int i;
    for(i = 0; i < 10; i++)
        scanf("%d",&a[i]);
    printf("\n");
    for(i = 0; i < 10; i++)
        printf("%d",a[i]);

    return 0;
}
```

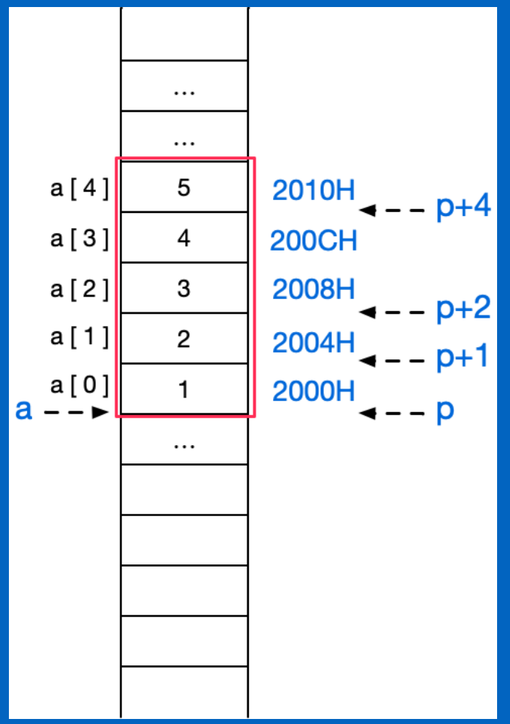
- 指针法

```
#include <stdio.h>
int main (){
    int a[10];
    int i;
    for(i = 0; i < 10; i++)
      scanf("%d",a + i);
    printf("\n");
    for(i= 0; i < 10; i++)
        printf("%d ",*(a + i));

    return 0;
}
```

© 东华大学

注意指针的当前值

```
#include <stdio.h>
int main (){
    int i,*p,a[5];
    p=a;
    for(i=0;i<5;i++)
        scanf("%d",p++);
    printf("\n");

    p = a;//如果漏掉会怎样?

    for(i=0;i<5;i++,p++)
        printf("%d\n",*p);

    return 0;
}
```

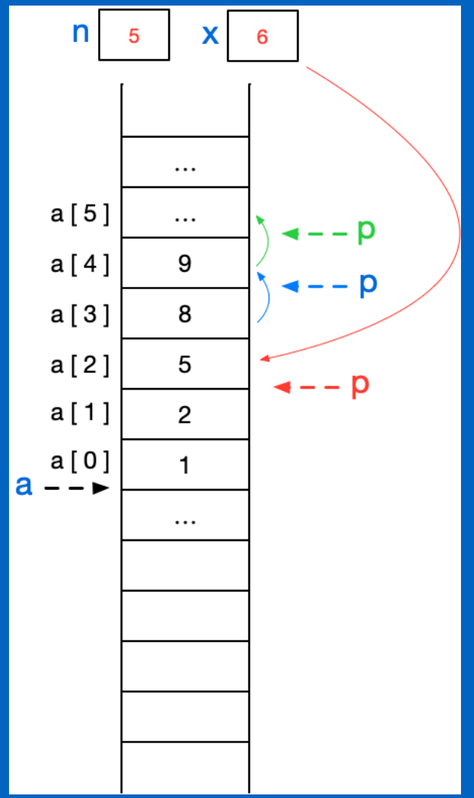


© 东华大学

数组插入元素（使用指针）

- 在有序数组中，插入一个新元素，使之仍然有序

```
#include <stdio.h>
#define MAX 100
int main (){
    int a[MAX],i,n,x,*p;
    // 初始化 a[],n, x, ...,n为数据个数, x为新元素的值

    for(p = a + n; (*(p - 1) > x) && (p > a); p--)
        *p = *(p - 1);

    *p = x;

    for(i = 0;i <= n; i++)
        printf("%3d",a[i]);
    printf("\n");

    return 0;
}
```

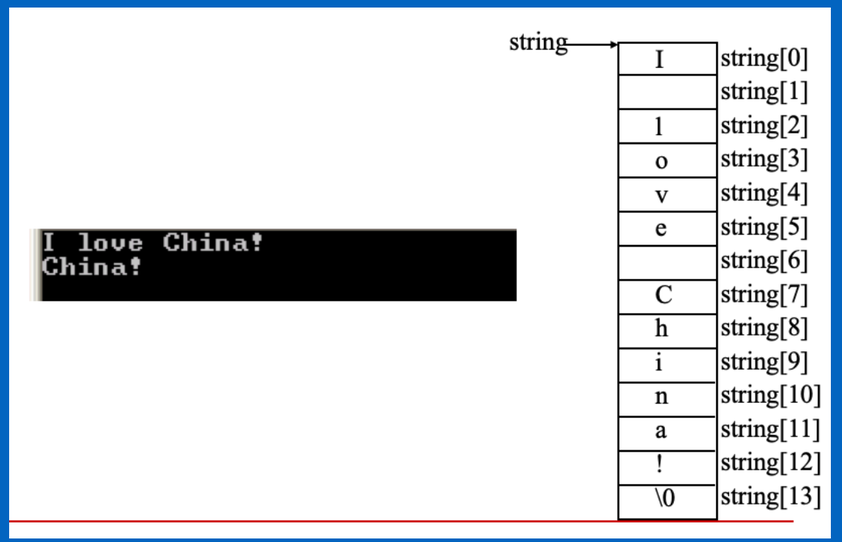


© 东华大学

字符串与指针

- 用字符数组实现
- 字符数组有若干个元素组成，每个元素中放一个字符
- 字符元素可以使用下标法和指针法表示
- 数组名代表数组的开始地址，其值不能改变

```
char string[]="I love China!";
printf("%s\n",string);
printf("%s\n",string + 7);
// printf("%s\n",string += 7); 出错
```

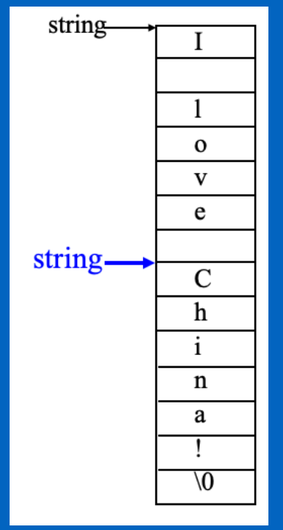


© 东华大学

字符串与指针

- 字符指针：指向一个字符的指针
- 字符指针中存放的是字符串的首地址(不是将整个字符串放到字符指针变量中)
- 指针变量的值可以改变

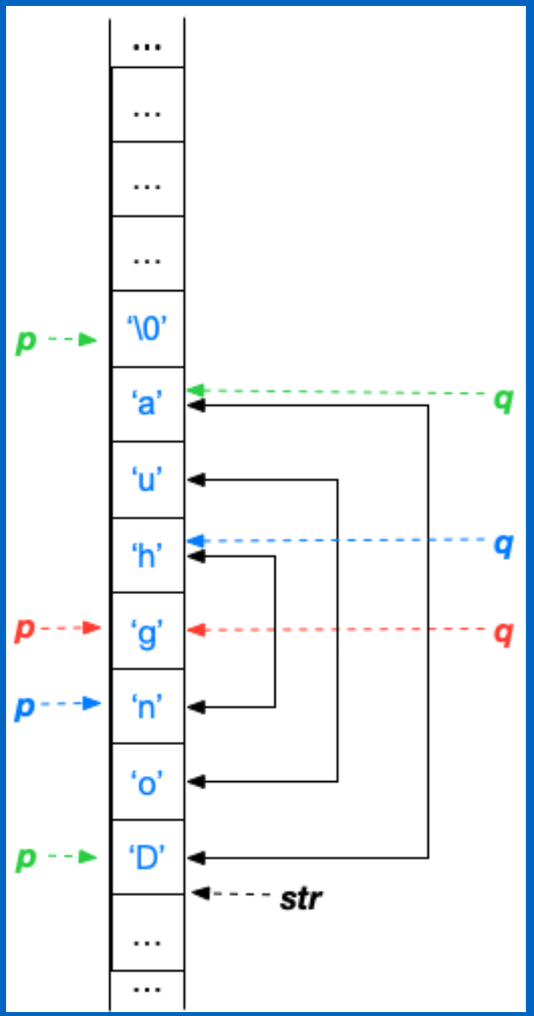
```
char  *string="I love China!";
printf("%s\n",string);
string += 7; //指针变量的值可以改变
while(*string != '\0'){
    putchar(string[0]);
    string++;
}
```



© 东华大学

字符串与指针

```
#include <stdio.h>
int main (){
    char *p,*q,ch,str[100];
    p = str;
    gets(p);

    while(*p != '\0')
        p++; // 寻找字符串最后一个字符的位置

    q = p-1;
    p = str;

    while(p < q){
        ch = *p;
        *p = *q;
        *q = ch;
        p++;
        q--;
    }

    puts(str);

    return 0;
}
```



© 东华大学

• 例: int a[]={1,2,3,4,5,6,7,8,9,10},*p=a,i; 则数组元素地址的正确表示：

A）&(a+1)
B）a++
C）&p
D）&p[i]

• 例:设有下列定义和语句

```
char str[20] = "Program",*p;
p = str;
```

则下列叙述中正确的是（ ）

A）*p 与str[0]的值相等
B）str 与p 的类型完全相同
C）str 数组长度和p 所指向的字符串长度相等
D）数组str 中存放的内容和指针变量p 中存放的内容相同

© 东华大学