

第8章 指针

——指针变量做函数参数

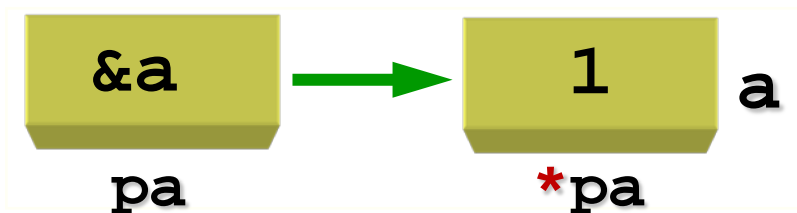
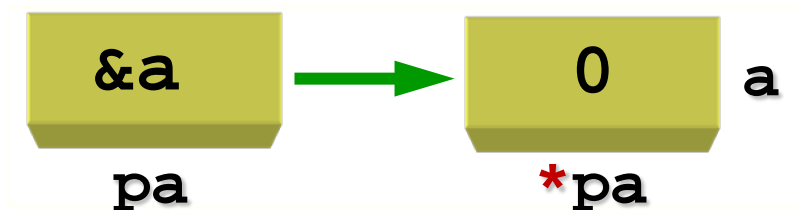
指针变量的解引用

■ 间接寻址运算符

* 此*非彼*

- 只要pa指向a，*pa就是a的别名

```
#include <stdio.h>
int main()
{
    int a = 0;
    int *pa = &a;
    *pa = 1;
    printf("a=%d\n", a);
    printf("*p=%d\n", *pa);
    return 0;
}
```

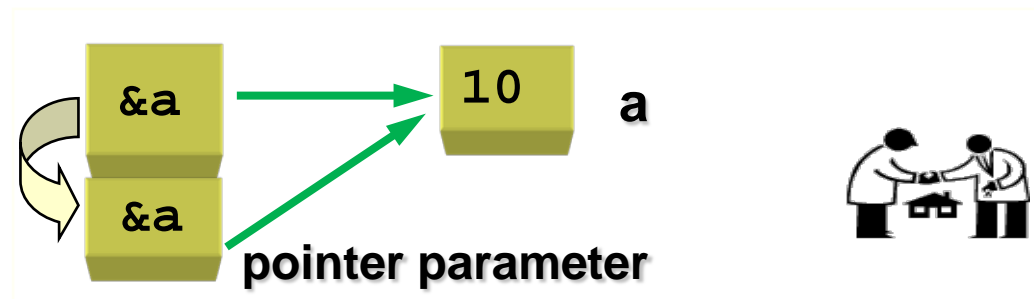
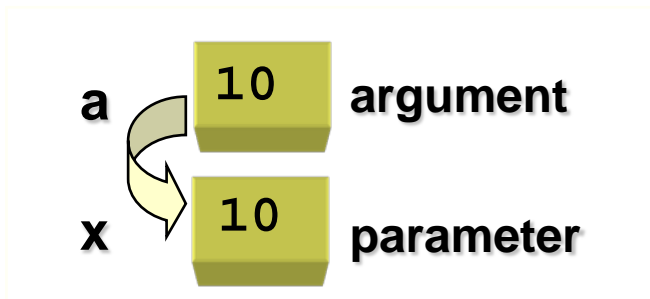


为什么要用指针?



为什么要用指针变量做函数参数？

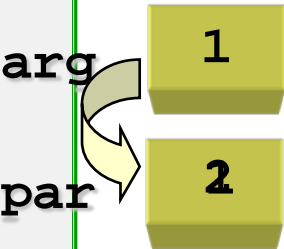
基本类型的变量作函数参数	指针类型的变量做函数参数
Call by Value——Passing arguments by value	Simulating Call by reference——Passing arguments by reference
实参变量的 值 → 形参 (parameter)	实参变量的 地址 → 指针 形参 (pointer parameter)
在被调函数中 不能 改变实参的值	在被调函数中 可以 改变实参的值



演示Call by value

```
#include <stdio.h>
void Fun(int par);
int main()
{
    int arg = 1 ;
    printf("arg = %d\n", arg);
    Fun(arg);
    printf("arg = %d\n", arg);
    return 0;
}
void Fun(int par)
{
    printf("par = %d\n", par);
    par = 2;
    return 0;
}
```

传实参变量的值



arg = 1
par = 1
arg = 1

形参值的改变
不影响对应的实参

```
#include <stdio.h>
int Fun(int par);
int main()
{
    int arg = 1 ;
    printf("arg = %d\n", arg);
    arg = Fun(arg);
    printf("arg = %d\n", arg);
    return 0;
}
int Fun(int par)
{
    printf("par = %d\n", par);
    par = 2;
    return par;
}
```

从函数返回修改的值

arg = 1
par = 1
arg = 2

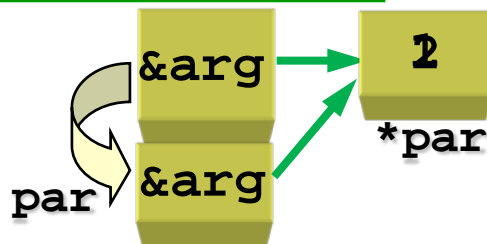
return仅限于
从函数返回一个值

演示Call by value

```
#include <stdio.h>
void Fun(int par);
int main()
{
    int arg = 1 ;
    printf("arg = %d\n", arg);
    Fun(arg);
    printf("arg = %d\n", arg);
    return 0;
}
void Fun(int par)
{
    printf("par = %d\n", par);
    par = 2;
    return 0;
}
```

传实参变量的值

arg = 1
par = 1
arg = 1



演示Simulating Call by reference

```
#include <stdio.h>
void Fun(int *par);
int main()
{
    int arg = 1 ;
    printf("arg = %d\n", arg);
    Fun(&arg);
    printf("arg = %d\n", arg);
    return 0;
}
void Fun(int *par)
{
    printf("par = %d\n", *par);
    *par = 2;
    return 0;
}
```

传实参变量的地址

arg = 1
par = 1
arg = 2

指针变量作形参为函数
提供了修改实参值的手段