哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

规格严格 功夫到家

第9章 指针

哈尔滨工业大学

计算机科学与技术学院

苏小红

sxh@hit.edu.cn

本章学习内容

- 指针数据类型
- 指针变量的定义和初始化
- 取地址运算符，间接寻址运算符
- 按值调用与按地址调用，指针变量作函数参数
- 函数指针

例7.9未能解决的问题：两数互换

程序 1：简单变量作函数参数

```
int main()
{
   int a, b;
   a = 5;
   b = 9;
   Swap(a, b);
   printf("a=%d,b=%d",a,b);
   return 0;
}
```

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

Not Work! Why?

程序 2：指针变量作函数参数

```
int main()
{
   int a,
   a = 5;
   b = 9;
   Swap( &a, &b );
   printf("a=%d,b=%d",a,b);
   return 0;
}
```

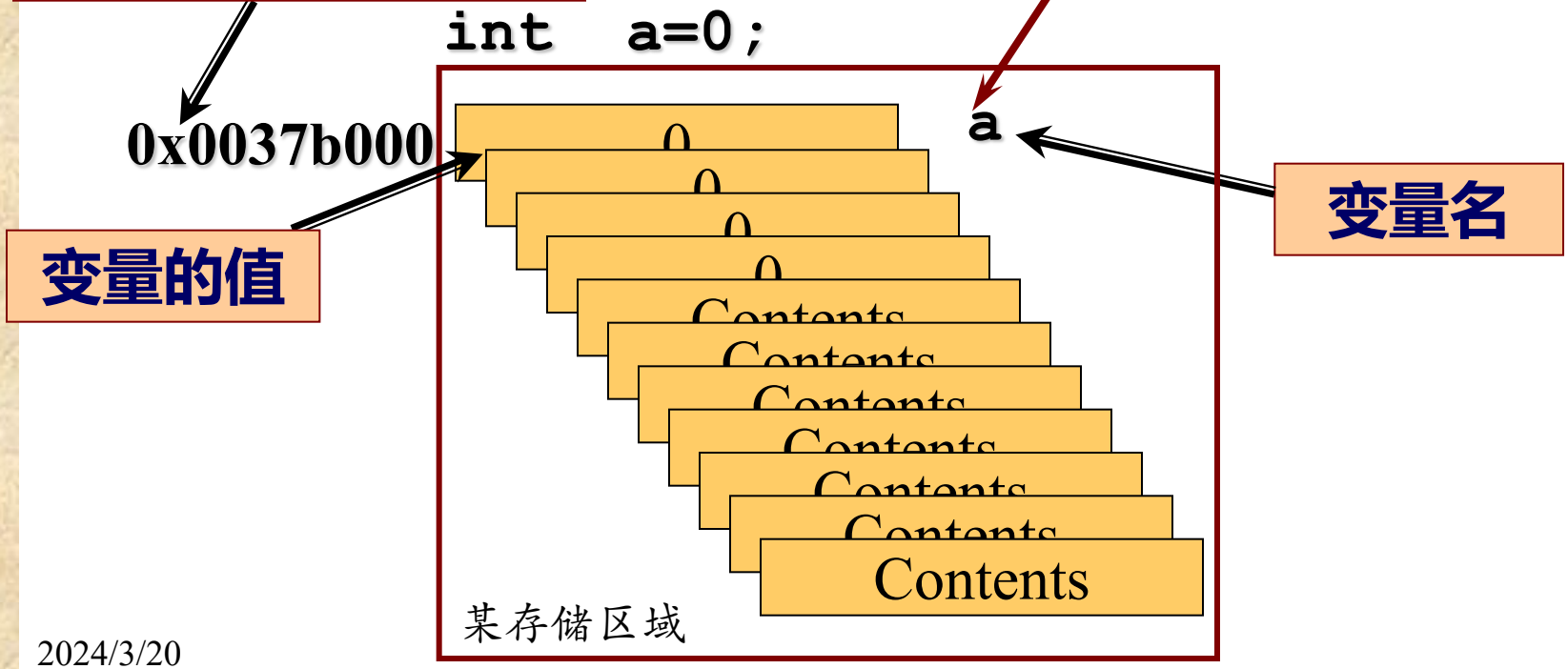
```
void Swap(int *x,int *y)
{
     int temp;
     temp = *x;
     *x = *y;
     *y = temp;
}
```

指针为函数提供修改变量值的手段

9.1变量的内存地址

内存：计算机内的存储部件
所有指令和数据都保存在内存里
速度快，可随机访问，但掉电即失
编译或函数调用时为变量分配内存单元

变量的地址(Address)

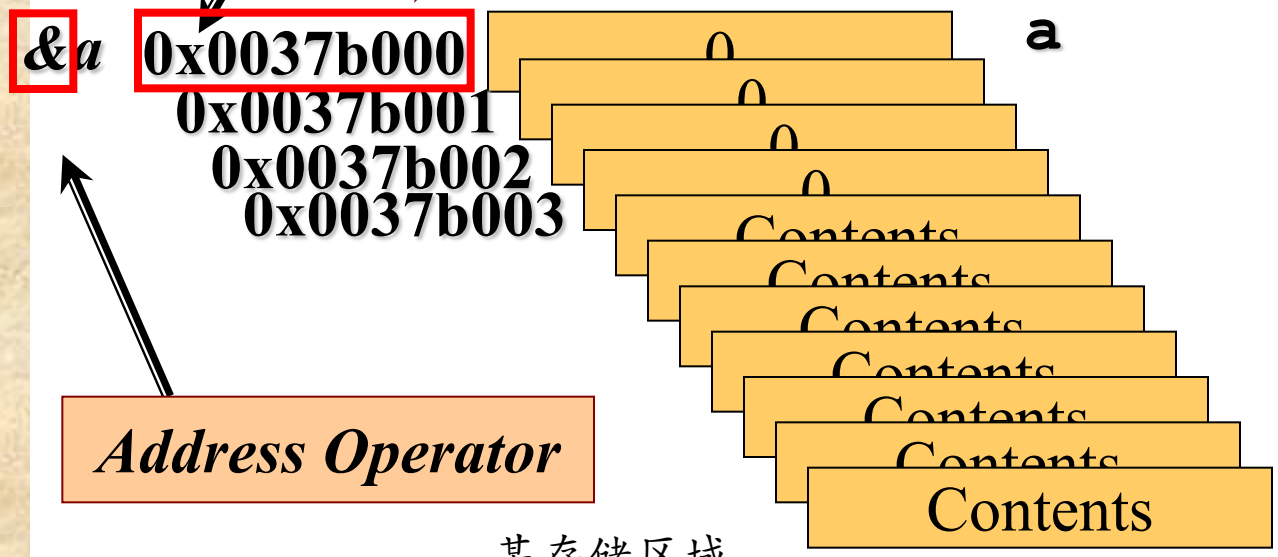


9.1变量的内存地址

内存中的每个字节都有唯一的编号（地址）
地址是一个十六进制无符号整数
其字长一般与主机相同
地址按字节编号，按类型分配空间

变量的地址

```
int  a=0;
```

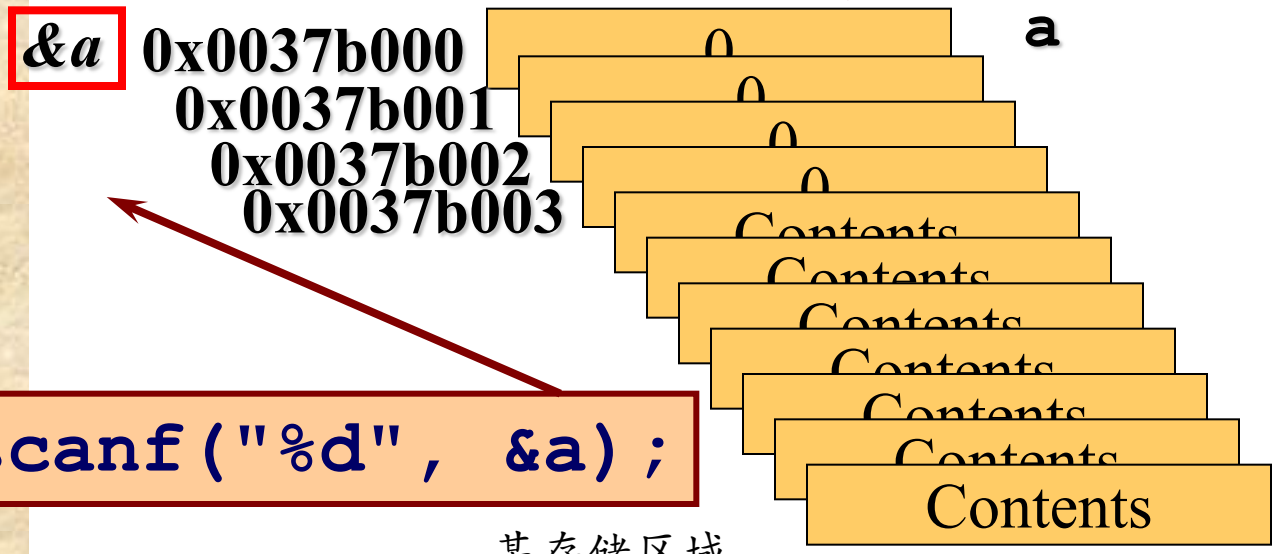


9.1变量的内存地址

只要指明要访问的变量的内存单元地址
就可以立即访问到变量所在的存储单元

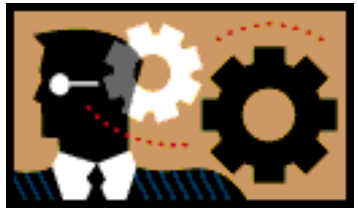
如何读写内存中的数据?

```
int  a=0;
```




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

某存储区域



9.1变量的内存地址

【例9.1】使用取地址运算符&取出变量的地址，然后将其显示在屏幕上。

```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       printf("a is %d, &a is %p\n", a, &a);
7       printf("b is %d, &b is %p\n", b, &b);
8       printf("c is %c, &c is %p\n", c, &c);
9       return 0;
10  }
```

表示输出变量a的地址值?

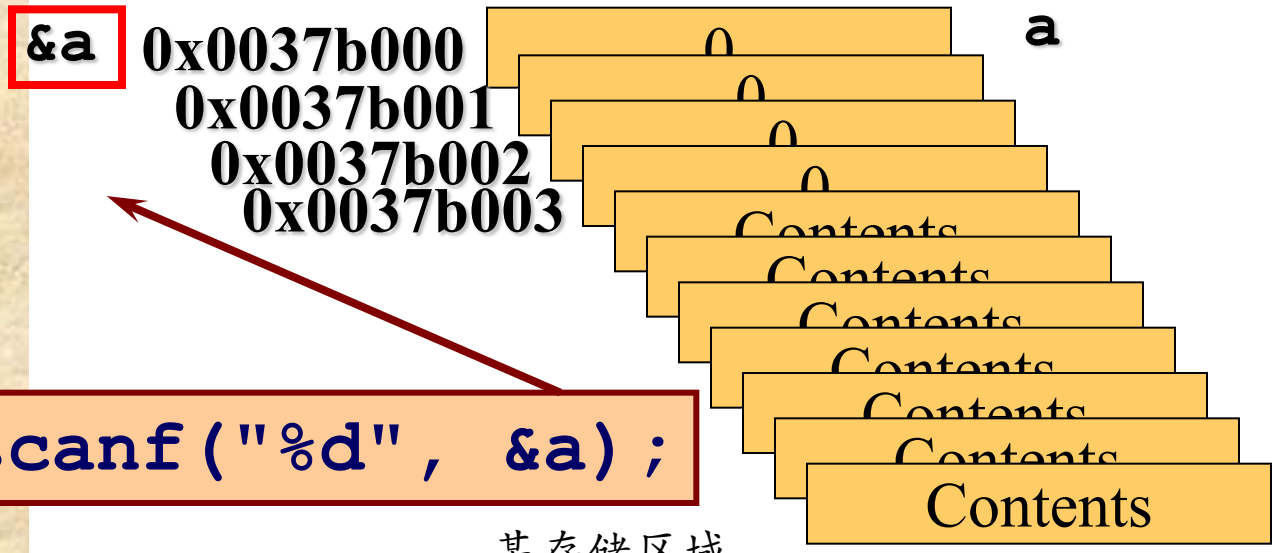
```
a is 0, &a is 0023FF74
b is 1, &b is 0023FF70
c is A, &c is 0023FF6F
```

9.1变量的内存地址

直接寻址：按变量地址存取变量值

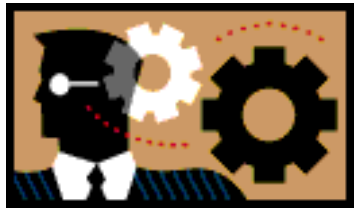
如何读写内存中的数据?

```
int  a=0;
```




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

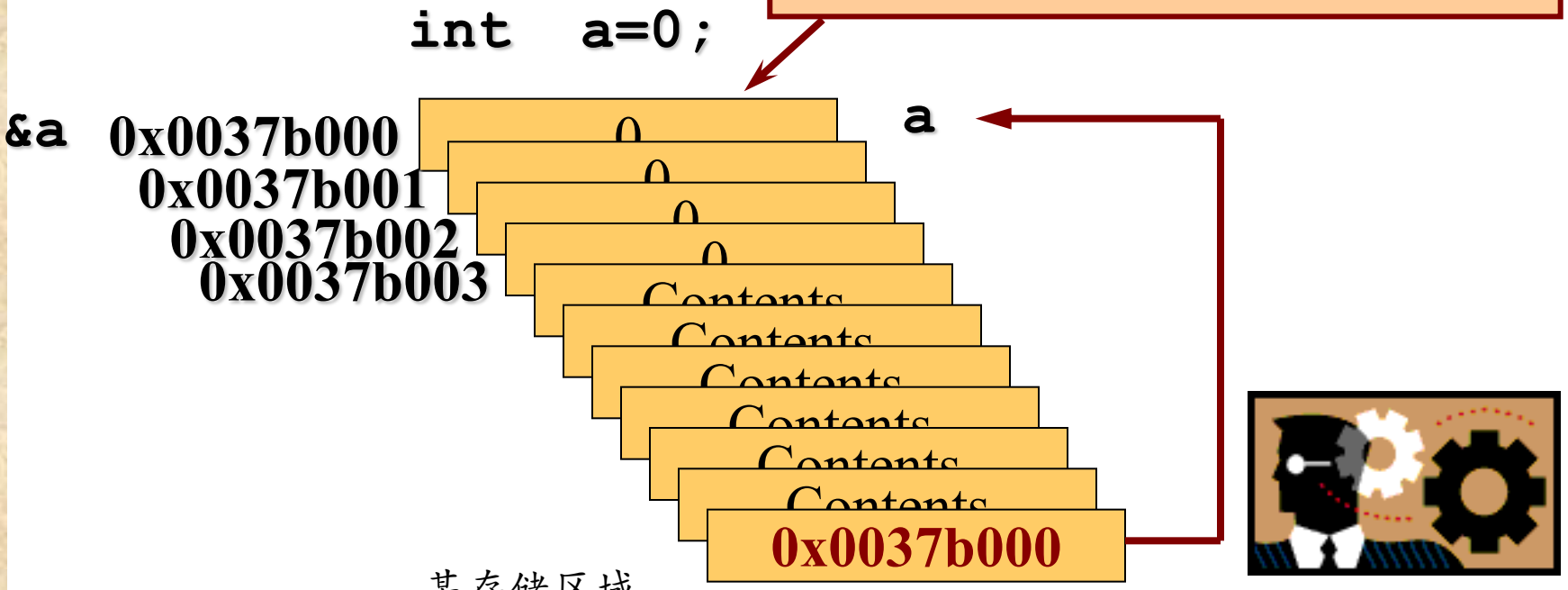
某存储区域



9.1变量的内存地址

间接寻址：通过存放变量地址的变量去访问变量

如何读写内存中的数据?



Errors


```
int i;
scanf("%d", i);
/* 这样会如何? */
```

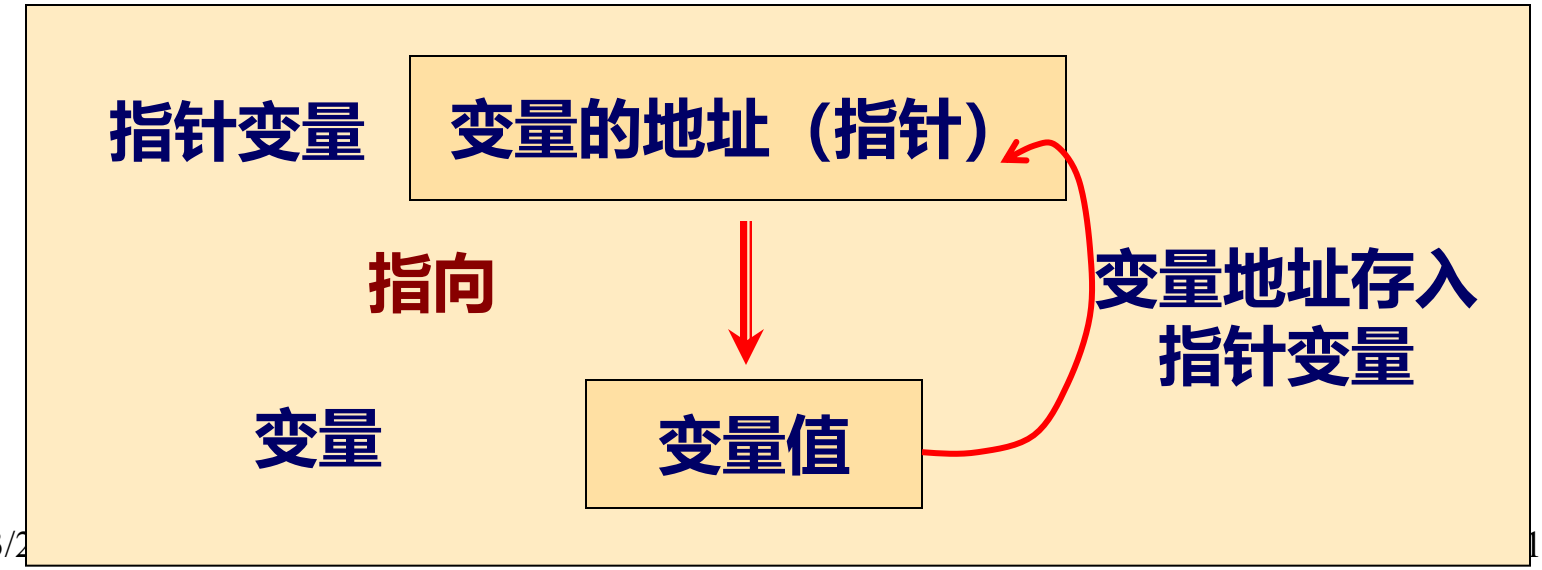
i的值被当作地址。如i==100，则输入的整数就会从地址100开始写入内存

```
char c;
scanf("%d", &c);
/* 这样呢? */
```

输入以int的二进制形式写到c所在的内存空间。
c所占内存不足以放下一个int，其后的空间也被覆盖

9.2指针变量的定义和初始化

- 存放变量的地址需要一种特殊类型的变量
- 指针（Pointer）类型
- 指针变量——具有指针类型的变量
- 变量的指针←→变量的地址



9.2指针变量的定义和初始化

【例9.2】使用指针变量在屏幕上显示变量的地址值

```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       int *pa, *pb;          /* 定义了可以指向整型数据的指针变量 pa 和 pb */
7       char *pc;              /* 定义了可以指向字符型数据的指针变量 pc */
8       printf("a is %d, &a is %p, pa is %p\n", a, &a, pa);
9       printf("b is %
10      printf("c is %
11      return 0;
12  }
```

定义了指针变量pa，但pa并未指向a?

如果指针指向一个非你控制的内存空间
并对该空间进行访问，将可能造成危险

```
a is 0, &a is 0023FF74, pa is 0023FF78
b is 1, &b is 0023FF70, pb is 00401394
c is A, &c is 0023FF6F, pc is 77C04E42
```

9.2指针变量的定义和初始化

【例9.2】使用指针变量在屏幕上显示变量的地址值

```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       int *pa, *pb;          /* 定义了可以指向整型数据的指针变量 pa 和 pb */
7       char *pc;              /* 定义了可以指向字符型数据的指针变量 pc */
8       printf("a is %d, &a is %p, pa is %p\n", a, &a, pa);
9       printf("b is %d, &b is %p, pb is %p \n", b, &b, pb);
10      printf("c is %c, &c is %p, pc is %p \n", c, &c, pc);
11      return 0;
12  }
```

指针变量使用之前必须初始化

Never use uninitialized pointers

```
warning: local variable 'pa' used without having been initialized
warning: local variable 'pb' used without having been initialized
warning: local variable 'pc' used without having been initialized
```

9.2指针变量的定义和初始化

【例9.2】使用指针变量在屏幕上显示变量的地址值

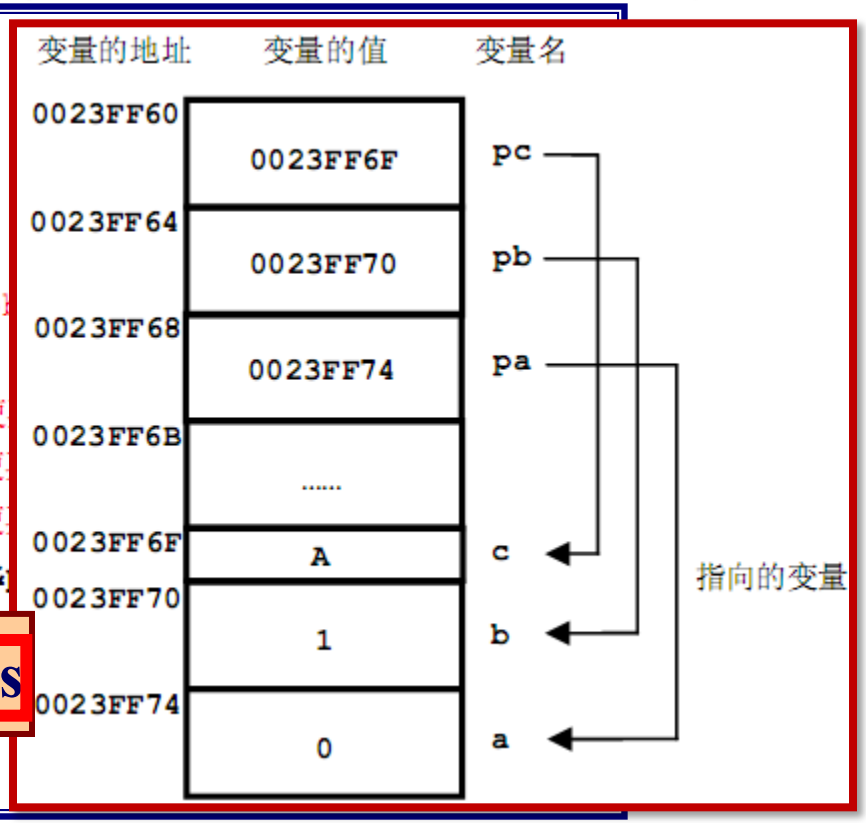
```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       int *pa = NULL, *pb = NULL; /* 定义指针变量并用 NULL 对其初始化 */
7       char *pc = NULL;            /* 定义指针变量并用 NULL 对其初始化 */
8       printf("a is %d, &a is %p, pa is %p\n", a, &a, pa);
9       printf("b is %d, &b is %p, pb is %p \n", b, &b, pb);
10      printf("c is %c, &c is %p, pc is %p \n", c, &c, pc);
11      return 0;
12  }
```

```
a is 0, &a is 0013FF7C, pa is 00000000
b is 1, &b is 0013FF78, pb is 00000000
c is A, &c is 0013FF74, pc is 00000000
```

9.2指针变量的定义和初始化

【例9.2】使用指针变量在屏幕上显示变量的地址值

```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       int  *pa, *pb;          /* 定义指针变量pa和p
7       char *pc;               /* 定义指针变量pc */
8       pa = &a;                /* 初始化指针变量pa使
9       pb = &b;                /* 初始化指针变量pb使
10      pc = &c;                /* 初始化指针变量pc使
11      printf("a is %d, &a is %p, pa is %p, &
14      return 0;
15  }
```



Pointers have names, types and values

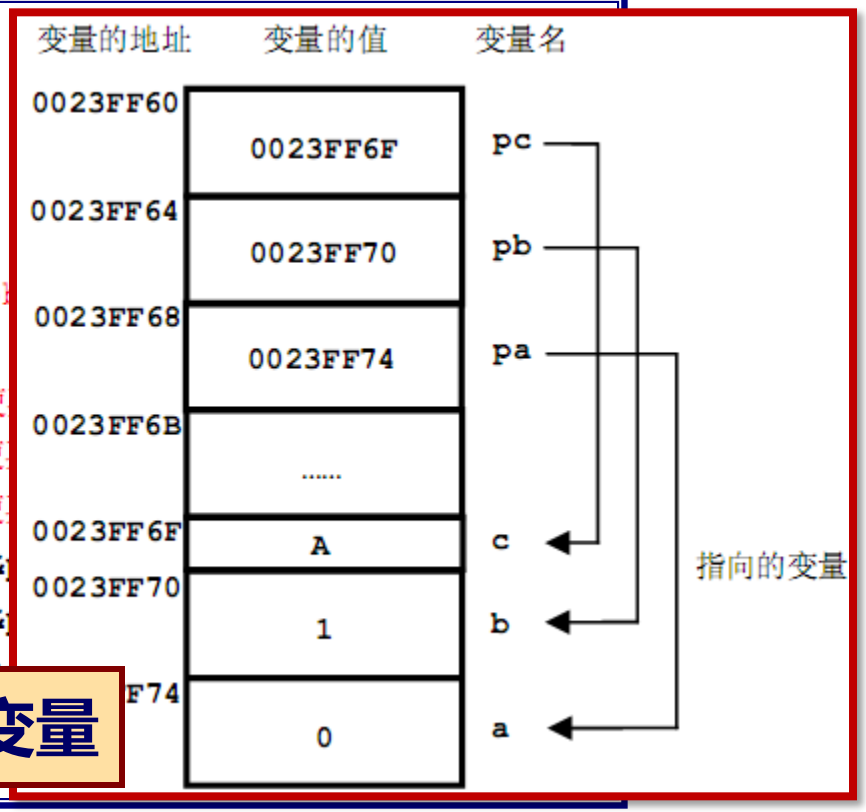
```
a is 0, &a is 0023FF74, pa is 0023FF74, &pa is 0023FF68
b is 1, &b is 0023FF70, pb is 0023FF70, &pb is 0023FF64
c is A, &c is 0023FF6F, pc is 0023FF6F, &pc is 0023FF60
```

9.2指针变量的定义和初始化

指针变量指向的数据类型称为基类型

显示变量的地址值

```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       int *pa, *pb;          /* 定义指针变量pa和p
7       char *pc;              /* 定义指针变量pc */
8       pa = &a;               /* 初始化指针变量pa使
9       pb = &b;               /* 初始化指针变量pb使
10      pc = &c;               /* 初始化指针变量pc使
11      printf("a is %d, &a is %p, pa is %p, &
12      printf("b is %d, &b is %p, pb is %p, &
```



指针变量只能指向同一基类型的变量

```
a is 0, &a is 0023FF74, pa is 0023FF74, &pa is 0023FF68
b is 1, &b is 0023FF70, pb is 0023FF70, &pb is 0023FF64
c is A, &c is 0023FF6F, pc is 0023FF6F, &pc is 0023FF60
```

9.2指针变量的定义和初始化

【例9.2】使用指针变量在屏幕上显示变量的地址值

```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       int *pa, *pb;          /* 定义指针变量pa和pb */
7       char *pc;              /* 定义指针变量pc */
8       pa = &a;               /* 初始化指针变量pa使其指向a */
9       pb = &b;               /* 初始化指针变量pb使其指向b */
10      pc = &c;               /* 初始化指针变量pc使其指向c */
11      printf("a is %d, &a is %p, pa is %p, &pa is %p\n", a, &a, pa, &pa);
12      printf("b is %d, &b is %p, pb is %p, &pb is %p\n", b, &b, pb, &pb);
13      printf("c is %c, &c is %p, pc is %p, &pc is %p\n", c, &c, pc, &pc);
14      return 0;
15  }
```

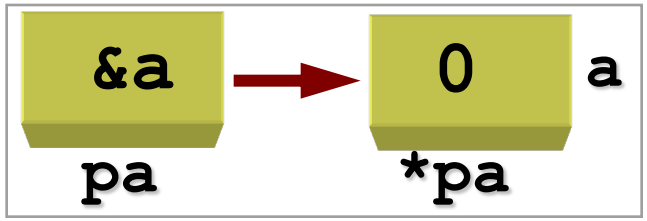
不能写成：int *pa, pb;

```
a is 0, &a is 0023FF74, pa is 0023FF74, &pa is 0023FF68
b is 1, &b is 0023FF70, pb is 0023FF70, &pb is 0023FF64
c is A, &c is 0023FF6F, pc is 0023FF6F, &pc is 0023FF60
```

9.3 间接寻址运算符

【例9.3】使用指针变量，通过间接寻址输出变量的值

```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       int *pa = &a, *pb = &b; /* 在定义指针变量 pa 和 pb 的同时对其初始化 */
7       char *pc = &c;          /* 在定义指针变量 pc 的同时对其初始化 */
8       printf("a is %d, &a is %p, pa is %p, *pa is %d\n", a, &a, pa, *pa);
9       printf("b is %d, &b is %p, pb is %p, *pb is %d\n", b, &b, pb, *pb);
10      printf("c is %c, &c is %p, pc is %p, *pc is %c\n", c, &c, pc, *pc);
11      return 0;
12  }
```

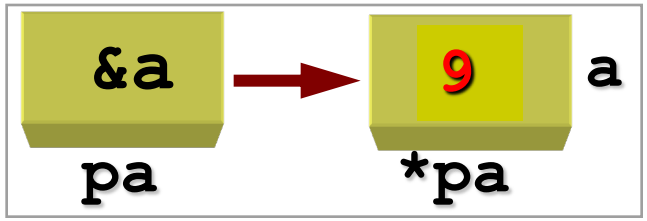



```
a is 0, &a is 0023FF74, pa is 0023FF74, *pa is 0
b is 1, &b is 0023FF70, pb is 0023FF70, *pb is 1
c is A, &c is 0023FF6F, pc is 0023FF6F, *pc is A
```

9.3 间接寻址运算符

【例9.3】使用指针变量，通过间接寻址输出变量的值

```
1   #include <stdio.h>
2   int main()
3   {
4       int  a = 0, b = 1;
5       char c = 'A';
6       int *pa = &a, *pb = &b; /* 在定义指针变量 pa 和 pb 的同时对其初始化 */
7       char *pc = &c;          /* 在定义指针变量 pc 的同时对其初始化 */
8       *pa = 9;                /* 修改指针变量 pa 所指向的变量的值 */
9       printf("a is %d, &a is %p, pa is %p, *pa is %d\n", a, &a, pa, *pa);
10      printf("b is %d, &b is %p, pb is %p, *pb is %d\n", b, &b, pb, *pb);
11      printf("c is %c,
12      return 0;
13  }
```



引用指针所指向的变量的值
称为指针的解引用(Pointer Dereference)

```
a is 9, &a is 0023FF74, pa is 0023FF74, *pa is 9
b is 1, &b is 0023FF70, pb is 0023FF70, *pb is 1
c is A, &c is 0023FF6F, pc is 0023FF6F, *pc is A
```

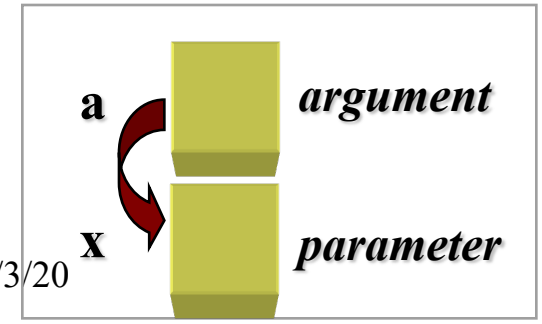
9.4 按值调用与按地址调用

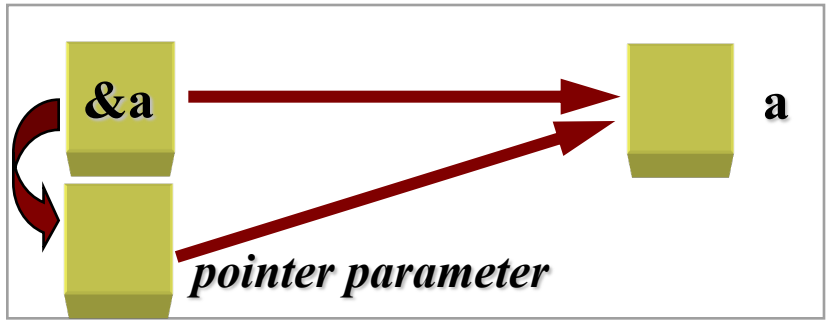
- 普通变量作函数参数—按值调用(*Call by Value*)
  - Can not modify the `argument`

  形参（*parameter*）← 实参变量（*variable*）

- 指针作函数参数—按地址调用(*Call by Reference*)
  - In order to modify the `argument`, use:

  指针形参(*pointer parameter*) ← *&(variable)*





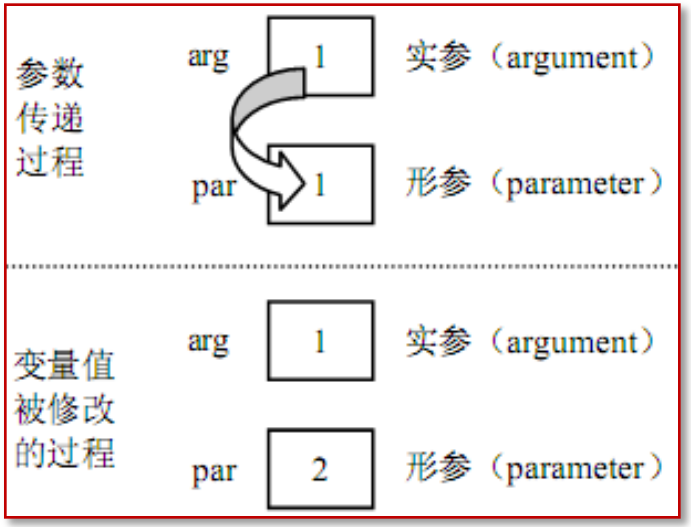
9.4 按值调用与按地址调用

【例9.4】演示按值调用

```
1    #include <stdio.h>
2    void Fun(int *par);
3    int main()
4    {
5        int arg = 1 ;
6        printf("arg = %d\n", arg);
7        Fun(arg);
8        printf("arg = %d\n", arg);
9        return 0;
10   }
11   void Fun(int par)
12   {
13       printf("par = %d\n", par);
14       par = 2;
15   }
```

传变量的值

```
arg = 1
par = 1
arg = 1
```

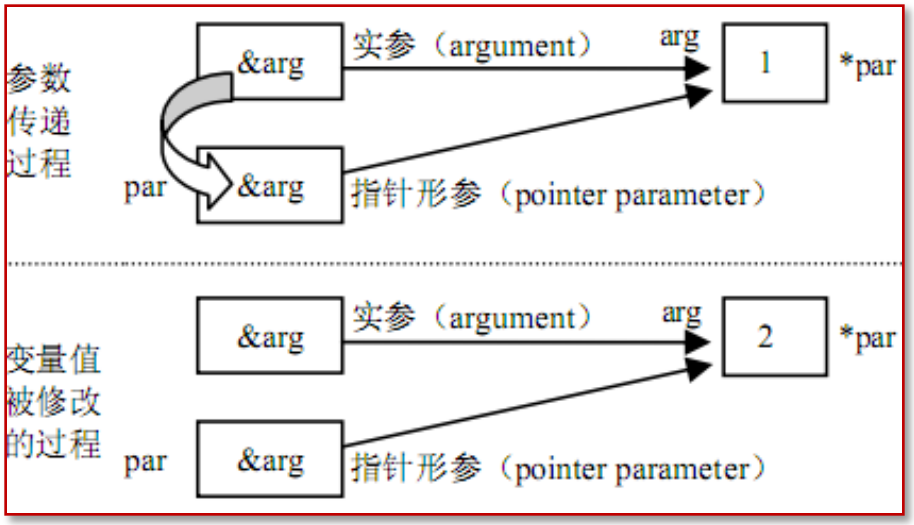


形参值的改变
不会影响对应的实参

9.4 按值调用与按地址调用

【例9.5】演示按地址调用

```
arg = 1
par = 1
arg = 2
```




```
1   #include <stdio.h>
2   void Fun(int *par);
3   int main()
4   {
5       int arg = 1 ;
6       printf("arg = %d\n", arg);
7       Fun(&arg);
8       printf("arg = %d\n", arg);
9       return 0;
10  }
11  void Fun(int *par)
12  {
13      printf("par = %d\n", *par);
14      *par = 2;
15  }
```

传变量地址

9.4 按值调用与按地址调用

【例9.4】演示按值调用

```
1    #include <stdio.h>
2    void Fun(int *par);
3    int main()
4    {
5        int arg = 1 ;
6        printf("arg = %d\n", arg);
7        Fun(arg);
8        printf("arg = %d\n", arg);
9        return 0;
10   }
11   void Fun(int par)
12   {
13       printf("par = %d\n", par);
14       par = 2;
15   }
```

```
arg = 1
par = 1
arg = 1
```

【例9.5】演示按地址调用

```
1    #include <stdio.h>
2    void Fun(int *par);
3    int main()
4    {
5        int arg = 1 ;
6        printf("arg = %d\n", arg);
7        Fun(&arg);
8        printf("arg = %d\n", arg);
9        return 0;
10   }
11   void Fun(int *par)
12   {
13       printf("par = %d\n", *par);
14       *par = 2;
15   }
```

指针变量作函数参数
可以修改实参的值

```
arg = 1
par = 1
arg = 2
```

9.4 按值调用与按地址调用

【例9.4】演示按值调用

```
1    #include <stdio.h>
2    void Fun(int *par);
3    int main()
4    {
5        int arg = 1 ;
6        printf("arg = %d\n", arg);
7        Fun(arg);
8        pr
9        re
10   }
11   void Fun(int par)
12   {
13       printf("par = %d\n", par);
14       par = 2;
15   }
```

```
arg = 1
par = 1
arg = 1
```

return仅限于从函数返回一个值

```
1    #include <stdio.h>
2    void Fun(int *par);
3    int main()
4    {
5        int arg = 1 ;
6        printf("arg = %d\n", arg);
7        arg = Fun(arg);
8        printf("arg = %d\n", arg);
9        return 0;
10   }
11   int Fun(int par)
12   {
13       printf("par = %d\n", par);
14       par = 2;
15       return par;
16   }
```

```
arg = 1
par = 1
arg = 2
```

例9.6：编写函数实现两数的互换

程序 1　　Trace the execution　　程序 2

```
int main()
{
   int a, b;
   a = 5;
   b = 9;
   Swap(a, b);
   printf("a=%d,b=%d",a,b);
   return 0;
}
```

主调函数

实 参

```
int main()
{
   int a, b;
   a = 5;
   b = 9;
   Swap( &a, &b );
   printf("a=%d,b=%d",a,b);
   return 0;
}
```

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

被调函数

形 参

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

Not Work! Why?

结果有何不同?

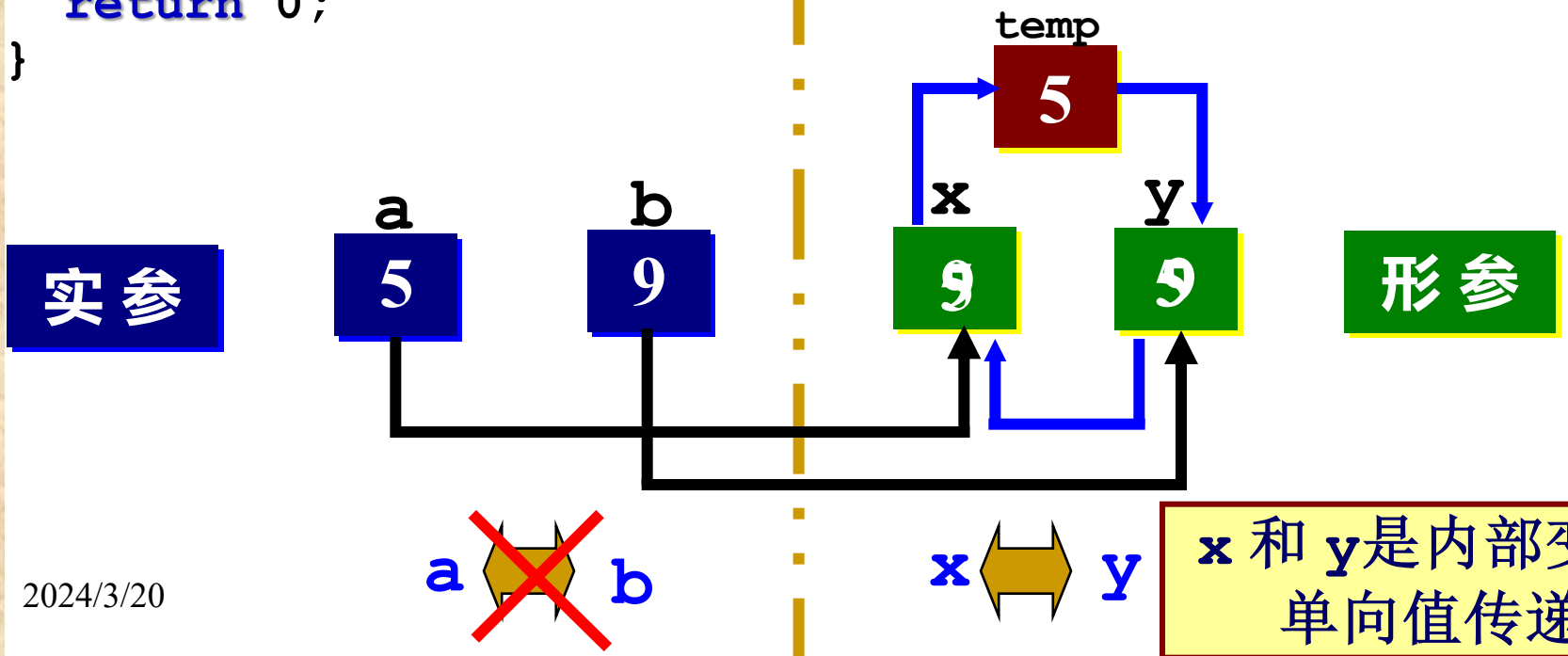
例9.6：编写函数实现两数的互换

主调函数

```
int main()
{
   int a, b;
   a = 5;
   b = 9;
   Swap(a, b);
   printf("a=%d,b=%d",a,b);
   return 0;
}
```

被调函数

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



x 和 y是内部变量
单向值传递

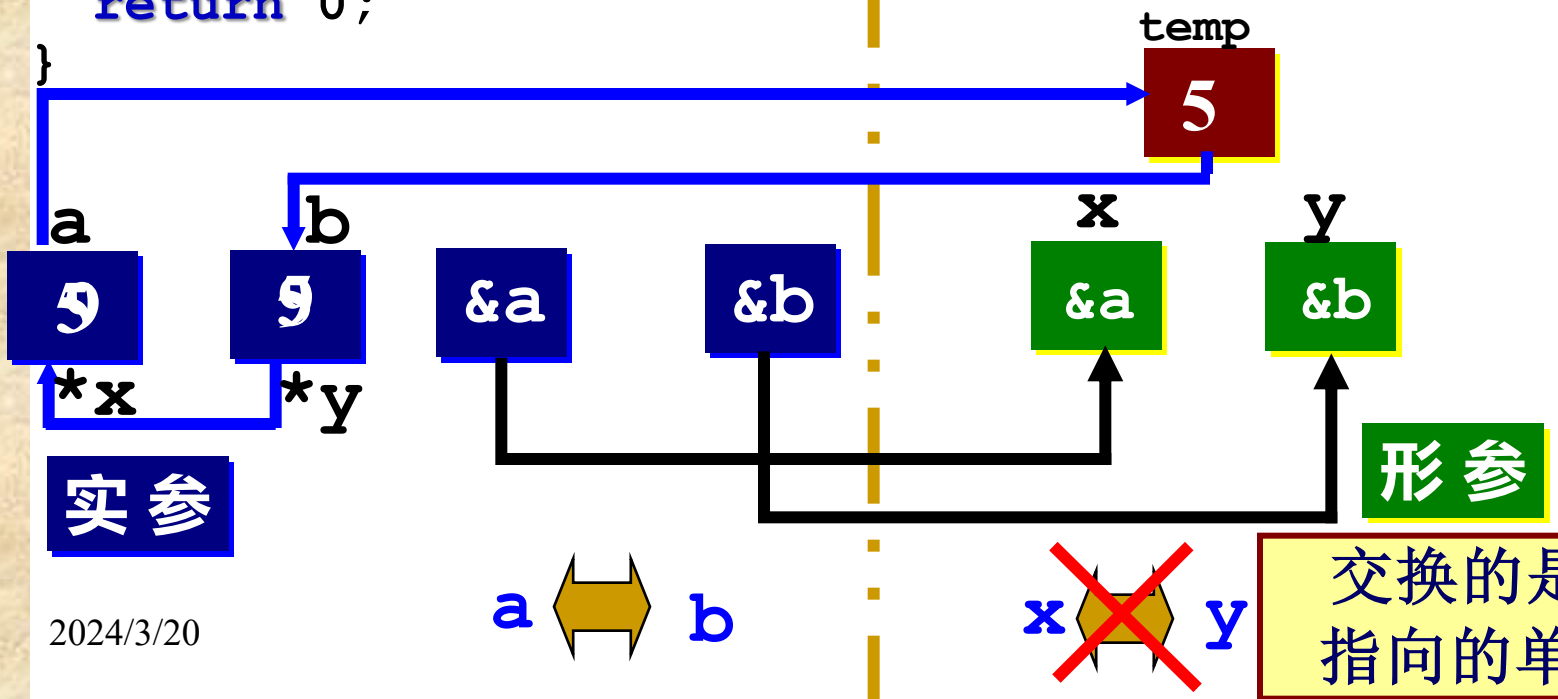
例9.6：编写函数实现两数的互换

主调函数

```
int main()
{
   int a, b;
   a = 5;
   b = 9;
   Swap( &a, &b );
   printf("a=%d,b=%d",a,b);
   return 0;
}
```

被调函数

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



Errors

```
void Swap(int *x, int *y)
{
    int *pTemp;

    *pTemp = *x;
    *x = *y;
    *y = *pTemp;
}
```

指针pTemp未初始化
指针pTemp指向哪里未知
对未知单元写操作是危险的

```
void Swap(int *x, int *y)
{
    int temp;

    temp = *x;
    *x = *y;
    *y = temp;
}
```

永远要清楚：
每个指针指向了哪里
指针指向的内容是什么

Errors

```
void Swap(int *x, int *y)
{
    int *pTemp;

    pTemp = x;
    x = y;
    y = pTemp;
}
```

指针pTemp被赋了值
但交换的是地址值
不是指针指向单元的内容

```
void Swap(int *x, int *y)
{
    int temp;

    temp = *x;
    *x = *y;
    *y = temp;
}
```

9.5用指针变量作函数参数的程序实例

- 【例9.7】计算并输出最高分及相应学生的学号

```
4   int main()
5   {
6       int  score[N], maxScore;
7       int    n, i;
8       long   num[N], maxNum;
9       printf("How many students?");
10      scanf("%d", &n);
11      printf("Input student's ID and score:\n");
12      for (i=0; i<n; i++)
13      {
14          scanf("%ld%d", &num[i], &score[i]);   /* 字母d前为字母l */
15      }
16      FindMax(score, num, n, maxScore, maxNum); /* 按值调用函数 */
17      printf("maxScore = %d, maxNum = %ld\n", maxScore, maxNum);
18      return 0;
19  }
```

9.5用指针变量作函数参数的程序实例

- 【例9.7】计算并输出最高分及相应学生的学号

```
/* 函数功能：计算最高分及其相应学生的学号 */
void FindMax(int score[],long num[],int n,int *pMaxScore,long *pMaxNum)
{
    int    i;
    *pMaxScore = score[0];              /* 假设score[0]为当前最高分 */
    *pMaxNum = num[0];                  /* 记录score[0]的学号num[0] */
    for (i=1; i<n; i++)                 /* 对所有score[i]进行比较 */
    {
        if (score[i] > *pMaxScore)      /* 如果score[i]高于当前最高分 */
        {
            *pMaxScore = score[i];      /* 用score[i]修改当前最高分 */
            *pMaxNum = num[i];          /* 记录当前最高分学生的学号num[i] */
        }
    }
}
```

```
How many students? 2↙
Input student's ID and score:
070310122   84↙
070310123   83↙
maxScore = -858993460, maxNum = -858993460
```

Not Work! Why?

9.5用指针变量作函数参数的程序实例

- 【例9.7】计算并输出最高分及相应学生的学号

```
4   int main()
5   {
6       int  score[
7       int    n, i
8       long   num[N], maxNum;
9       printf("How many students?");
10      scanf("%d", &n);
11      printf("Input student's ID and score:\n");
12      for (i=0; i<n; i++)
13      {
14          scanf("%ld%d", &num[i], &score[i]);   /* 字母d前为字母l */
15      }
16      FindMax(score, num, n, maxScore, maxNum); /* 按值调用函数 */
17      printf("maxScore = %d, maxNum = %ld\n", maxScore, maxNum);
18      return 0;
19  }
```

真正的原因：普通变量作函数参数按值调用不能在被调函数中改变相应的实参值

```
warning: local variable 'maxNum' used without having been initialized
warning: local variable 'maxScore' used without having been initialized
```

9.5用指针变量作函数参数的程序实例

- 【例9.7】计算并输出最高分及相应学生的学号

```
4   int main()
5   {
6       int  score[N], maxScore;
7       int    n, i;
8       long   num[N], maxNum;
9       printf("How many students?");
10      scanf("%d", &n);
11      printf("Input student's ID and score:\n");
12      for (i=0; i<n; i++)
13      {
14          scanf("%ld%d", &num[i], &score[i]);   /* 字母d前为字母l */
15      }
16      FindMax(score, num,n, &maxScore, &maxNum);/* 按地址调用函数 */
17      printf("maxScore = %d, maxNum = %ld\n", maxScore, maxNum);
18      return 0;
19  }
```

9.5用指针变量作函数参数的程序实例

- 【例9.7】计算并输出最高分及相应学生的学号

```
20 /* 函数功能：计算最高分及其相应学生的学号 */
21 void FindMax(int score[],long num[],int n,int *pMaxScore,long *pMaxNum)
22 {
23     int   i;
24     *pMaxScore = score[0];          /* 假设score[0]为当前最高分 */
25     *pMaxNum = num[0];              /* 记录score[0]的学号num[0] */
26     for (i=1; i<n; i++)             /* 对所有score[i]进行比较 */
27     {
28         if (score[i] > *pMaxScore) /* 如果score[i]高于当前最高分 */
29         {
30             *pMaxScore = score[i];
31             *pMaxNum = num[i];
32         }
33     }
34 }
```

```
How many students? 5↙
Input student's ID and score:
070310122   84↙
070310123   83↙
070310124   88↙
070310125   87↙
070310126   61↙
maxScore = 88, maxNum = 070310124
```

9.6函数指针及其应用

- 函数指针(Function Pointers)就是指向函数的指针（Pointer to a Function）
- 指向函数的指针变量存储的是函数在内存中的入口地址
- 编译器将不带()的函数名解释为该函数的入口地址

  数据类型 (* 指针名)();

  - 例如：`int (*p)();`
- 常见错误：
  - 忘记了前一个()，写成
    - `int *p();` /*声明一个函数名为p、返回值是整型指针的函数*/
  - 忘掉了后一个()，写成
    - `int (*p);` /*定义了一个整型指针*/
  - 定义时后一个括号内的参数类型与指向的函数参数类型不匹配

9.6函数指针及其应用

- 应用
  - 编写通用性更强的函数
- 典型实例1
  - 计算函数的定积分
- 典型实例2
  - 既能按照升序排序，又能按照降序排序

9.6函数指针及其应用

- 【例9.8】修改例8.8中的排序函数，使其既能实现对学生成绩的升序排序，又能实现对学生成绩的降序排序
- **先不使用函数指针编程**

```
1    #include <stdio.h>
2    #define N 40
3    int ReadScore(int score[]);                    /* 成绩输入函数原型 */
4    void PrintScore(int score[], int n);           /* 成绩输出函数原型 */
5    void AscendingSort(int a[], int n);            /* 升序排序原函数型 */
6    void DescendingSort(int a[], int n);           /* 降序排序原函数型 */
```

应用

```c
#include <stdio.h>
#define N 40
int ReadScore(int score[]);                 /* 成绩输入函数原型 */
void PrintScore(int score[], int n);        /* 成绩输出函数原型 */
void AscendingSort(int a[], int n);         /* 升序排序函数原型 */
void DescendingSort(int a[], int n);        /* 降序排序函数原型 */
void Swap(int *x, int *y);                  /* 两数交换函数原型 */
int main()
{
   int score[N], n;
   int order;                       /* 值为1表示升序排序，值为2表示降序排序 */
   n = ReadScore(score);            /* 输入成绩，返回学生人数 */
   printf("Total students are %d\n",n);
   printf("Enter 1 to sort in ascending order,\n");
   printf("Enter 2 to sort in descending order:");
   scanf("%d", &order);
   printf("Data items in original order\n");
   PrintScore(score, n);            /* 输出排序前的成绩 */
   if (order == 1)
   {
        AscendingSort(score, n);  /* 按升序排序 */
        printf("Data items in ascending order\n");
   }
   else
   {
        DescendingSort(score, n); /* 按降序排序 */
        printf("Data items in descending order\n");
   }
   PrintScore(score, n);            /* 输出排序后的成绩 */
   return 0;
}
```

```
Input score:84 ↙
Input score:83 ↙
Input score:88 ↙
Input score:87 ↙
Input score:61 ↙
Input score:-1 ↙
Total students are 5
Enter 1 to sort in ascending order,
Enter 2 to sort in descending order:1↙
Data items in original order
   84  83  88  87  61
Data items in ascending order
    61  83  84  87  88
```

```
32    /* 函数功能：输入学生某门课的成绩，当输入负值时，结束输入，返回学生人数 */
33    int ReadScore(int score[])
34    {
35        int i = -1;
36        do{
37            i++;
38            printf("Input score:");
39            scanf("%d", &score[i]);
40        } while (score[i] >= 0);
41        return i;
42    }
43    /* 函数功能：输出学生成绩 */
44    void PrintScore(int score[], int n)
45    {
46        int i;
47        for (i=0; i<n; i++)
48        {
49            printf("%4d", score[i]);
50        }
51        printf("\n");
52    }
```

```
53  /* 函数功能：选择法实现数组a的升序排序 */
54  void AscendingSort(int a[], int n)
55  {
56      int i, j, k;
57      for (i=0; i<n-1; i++)
58      {
59          k = i;
60          for (j=i+1; j<n; j++)
61          {
62              if (a[j] < a[k])    k = j;
63          }
64          if (k != i)    Swap(&a[k], &a[i]);
65      }
66  }
```

```
67  /* 函数功能：选择法实现数组a的降序排序 */
68  void DescendingSort(int a[], int n)
69  {
70      int i, j, k;
71      for (i=0; i<n-1; i++)
72      {
73          k = i;
74          for (j=i+1; j<n; j++)
75          {
76              if (a[j] > a[k])    k = j;
77          }
78          if (k != i)    Swap(&a[k], &a[i]);
79      }
80  }
```

9.6函数指针及其应用

- 【例9.9】修改例9.8中的程序实例，用函数指针编程实现一个通用的排序函数，对学生成绩既能实现升序排序，又能实现降序排序
- **使用函数指针编程**

```
1    #include <stdio.h>
2    #define N 40
3    int ReadScore(int score[]);
4    void PrintScore(int score[], int n);
5    void SelectionSort(int a[], int n, int (*compare)(int a, int b));
6    int Ascending( int a, int b );
7    int Descending( int a, int b );
8    void Swap(int *x, int *y);
```

```
9    int main()
10   {
11      int score[N], n;
12      int order;                          /* 值为 1 表示升序排序，值为 2 表示降序排序 */
13      n = ReadScore(score);               /* 输入成绩，返回学生人数 */
14      printf("Total students are %d\n",n);
15      printf("Enter 1 to sort in ascending order,\n");
16      printf("Enter 2 to sort in descending order:");
17      scanf("%d", &order);
18      printf("Data items in original order\n");
19      PrintScore(score, n);               /* 输出排序前的成绩 */
20      if (order == 1)
21      {
22           SelectionSort(score, n, Ascending);/*函数指针指向 Ascending()*/
23           printf("Data items in ascending order\n");
24      }
25      else
26      {
27           SelectionSort(score, n, Descending);/*函数指针指向 Descending()*/
28           printf("Data items in descending order\n");
29      }
30      PrintScore(score, n);               /* 输出排序后的成绩 */
31      return 0;
32   }
```

9.6函数指针及其应用

```
54  /* 函数功能：调用函数指针compare指向的函数实现对数组a的交换法排序 */
55  void SelectionSort(int a[], int n, int (*compare)(int a, int b))
56  {
57      int i, j, k;
58      for (i=0; i<n-1; i++)
59      {
60          k = i;
61          for (j=i+1; j<n; j++)
62          {
63              if ((*compare)(a[j], a[k]))  k = j;
64          }
65          if (k != i)     Swap(&a[k], &a[i]);
66      }
67  }
```

9.6函数指针及其应用

```
void SelectionSort(int a[], int n,
                   int (*compare)(int a, int b))
{
   ……
   if ((*compare)(a[j], a[k]))
   ……
}
/*决定数据是否按升序排序,a<b为真,则按升序排序*/
int Ascending(int a, int b)
{
      return a < b;
}
/*决定数据是否按降序排序,a>b为真,则按降序排序*/
int Descending(int a, int b)
{
      return a > b;
}
```

指针变量与其他类型变量的对比

- 共性
  - 在内存中占据一定大小的存储单元
  - 先定义，后使用
- 特殊性
  - 它的内容只能是地址
  - 必须初始化后才能使用，否则指向不确定的存储单元
  - 只能指向同一基类型的变量
  - 可参与的运算：加、减整数，自增、自减、关系、赋值
- 使用原则
  - 明确指针指向了哪里
  - 明确指针指向单元的内容是什么
  - 永远不要使用未初始化的指针变量

Questions and answers