函数

本讲主要内容

- 函数的定义与声明
- 标准库函数
- 强制类型转换

函数

- 函数定义的一般

返回值数据类型　函数名　（数据类型 形参1，数据类型 形参2，…，数据类型 形参n）

函数头

函数体

```
{
    变量声明，如有变量;
    其他语句;
    return （ 表达式 ）;
}
```

➢可为**void**，表明无返回数据，此时函数体不应包含return语句。
➢缺省时，表明返回值为int。

形参可缺省，缺省时为*无参函数*。也可用**void**表明形参缺省。

return　表达式;
传递函数的返回值

函数的定义与声明
--函数的定义

```
float add ( float num1, float num2 )
{
    float sum;
    sum=num1+num2;
    return sum;
}
```

函数的定义与声明
--函数的定义

```
include <math.h>
double  Pi ( void )
{
    return  2*asin(1.0);
}
```

函数的定义与声明
--函数的定义

```
include <stdio.h>
void  displayChars ( char s,  unsigned int qty )
{
    unsigned int  i;
   printf("\n");
   for (i=0,  i<qty,  ++i)
        printf("%c", s);
    printf("\n");
}
```

函数的定义与声明
--函数的声明

- 函数原形的一般形式： function prototype
- *返回值数据类型 函数名 （数据类型 参数1，数据类型 参数2，…，数据类型 参数n）；*
- *返回值数据类型 函数名 （数据类型，数据类型，…，数据类型 ）；* ✓常用
- 函数原形中的参数列表必需与函数定义中的形参列表保持一致：个数、顺序及各参数数据类型。

函数的定义与声明
--函数的声明

```
float add ( float num1, float num2 )
{
   float sum;
   sum=num1+num2;
   return sum;
}
```

函数原形：

```
float add ( float      , float      ) ;
```

函数的定义与声明
——函数的调用

主调函数

- 函数[illegible]而先定义或声明后方可使用。

```
#include <stdio.h>
  float findMax(float, float);
  int main()
  {  float firstnum, secnum, maxnum;
     printf("Enter a number: ");
     scanf("%f", &firstnum);
     printf("\nGreat! Please enter a second number: ");
     scanf("%f", &secnum);
     maxnum = findMax(firstnum, secnum);
     printf("\nThe maximum of the two numbers entered is %f.\n",
maxnum);
     return 0;
  }
```

函数原形

被调函数

函数调用

实参列表

函数的定义与声明
——函数的调用

函数定义

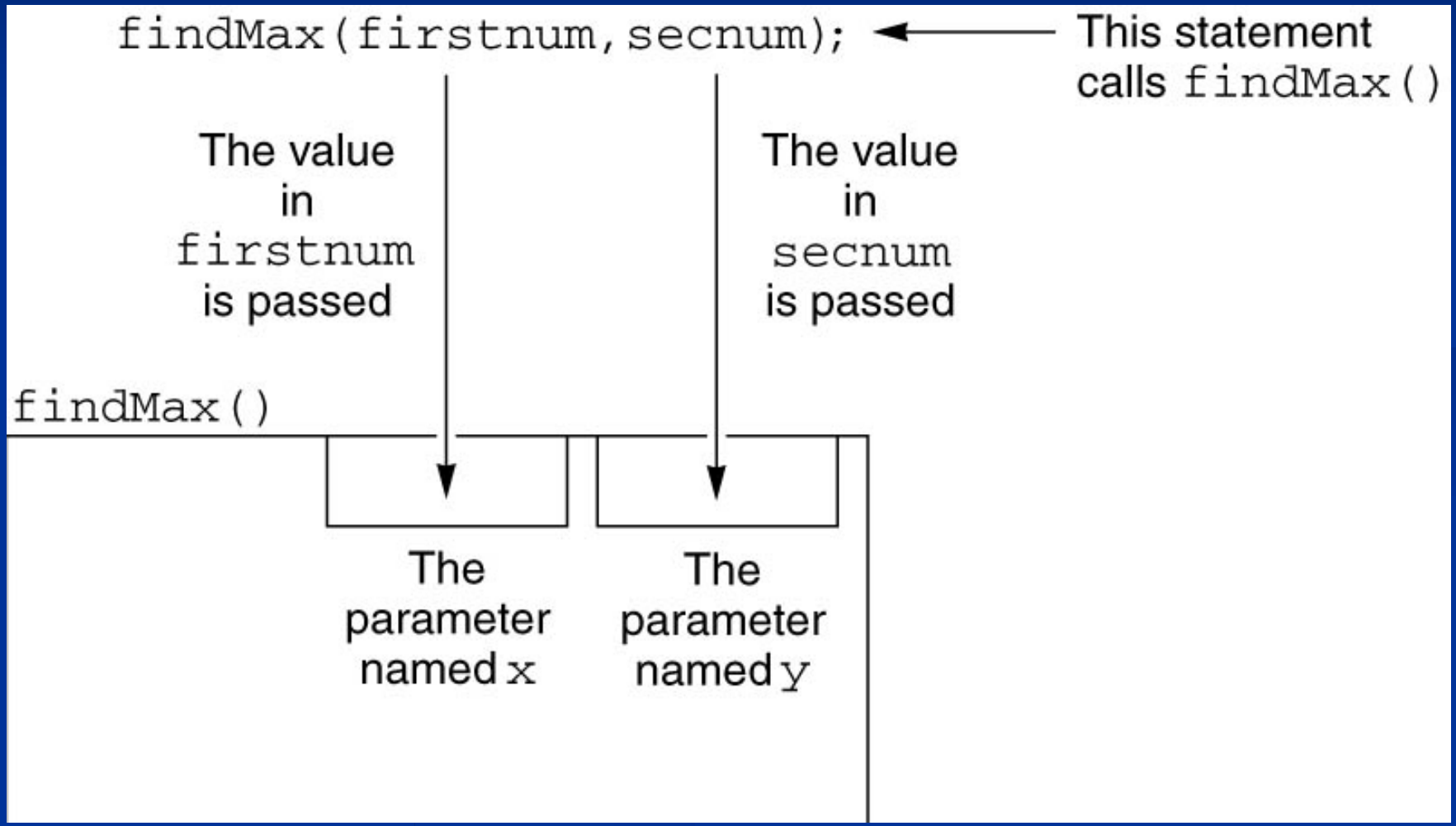
形参列表

```
/* the following is the function findMax  */

float  findMax ( float x, float y ) /* function header          */
{                                   /* start of function body  */
     float maxnum;                    /* variable declaration    */

     if (x >= y)                 /* find the maximum number    */
          maxnum = x;
     else
          maxnum = y;

     return ( maxnum );              /* return the value        */
}
```

返回值

函数的定义与声明
--函数的调用

- 函数的传值调用：实参向形参进行单向的数值传递。



Storing values into parameters

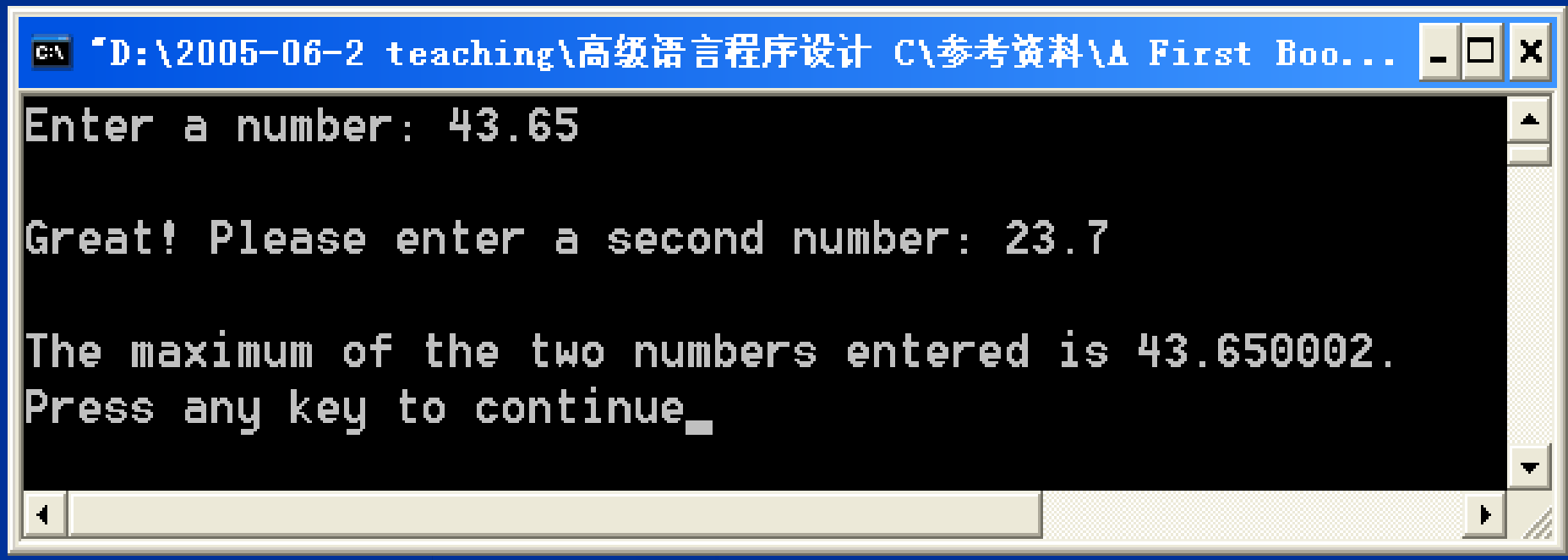
函数的定义与声明
--函数的调用

➢函数调用的执行：

```
#include <stdio.h>
  float   findMax(float, float);
  int main()
  {  float firstnum, secnum,
maxnum;
      printf("Enter a number: ");
      scanf("%f", &firstnum);
      printf("\nGreat! Please enter a
second number: ");
      scanf("%f", &secnum);
      maxnum = findMax(firstnum,
secnum);
      printf("\nThe maximum of the
two numbers entered is %f.\n",
maxnum);
      return 0;
   }
```

```
 /* the following is the
function findMax  */

float  findMax ( float x,
float y )
{
      float maxnum;

      if (x >= y)
            maxnum = x;
      else
            maxnum = y;

      return ( maxnum );
}
```

函数的定义与声明
--函数的调用



函数的定义与声明
--函数的调用

函数调用的三种方式：

按照函数在程序中出现的位置来分；

- 作为表达式出现在任何允许表达式出现的地方，参与运算。

  如：a＝sqrt(b)；

- 作为一条独立的语句完成特定的操作。

  如：gets(string1)；

- 作为函数的参数被其他函数调用。

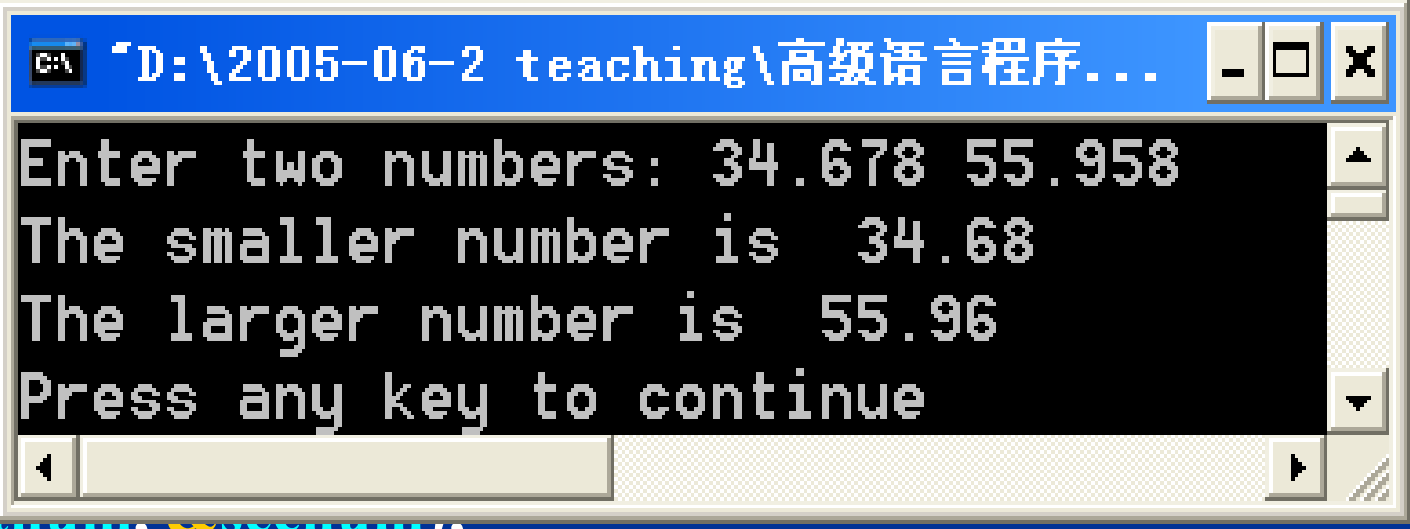
  如： h=sqrt(abs(theta))；

函数的传址调用

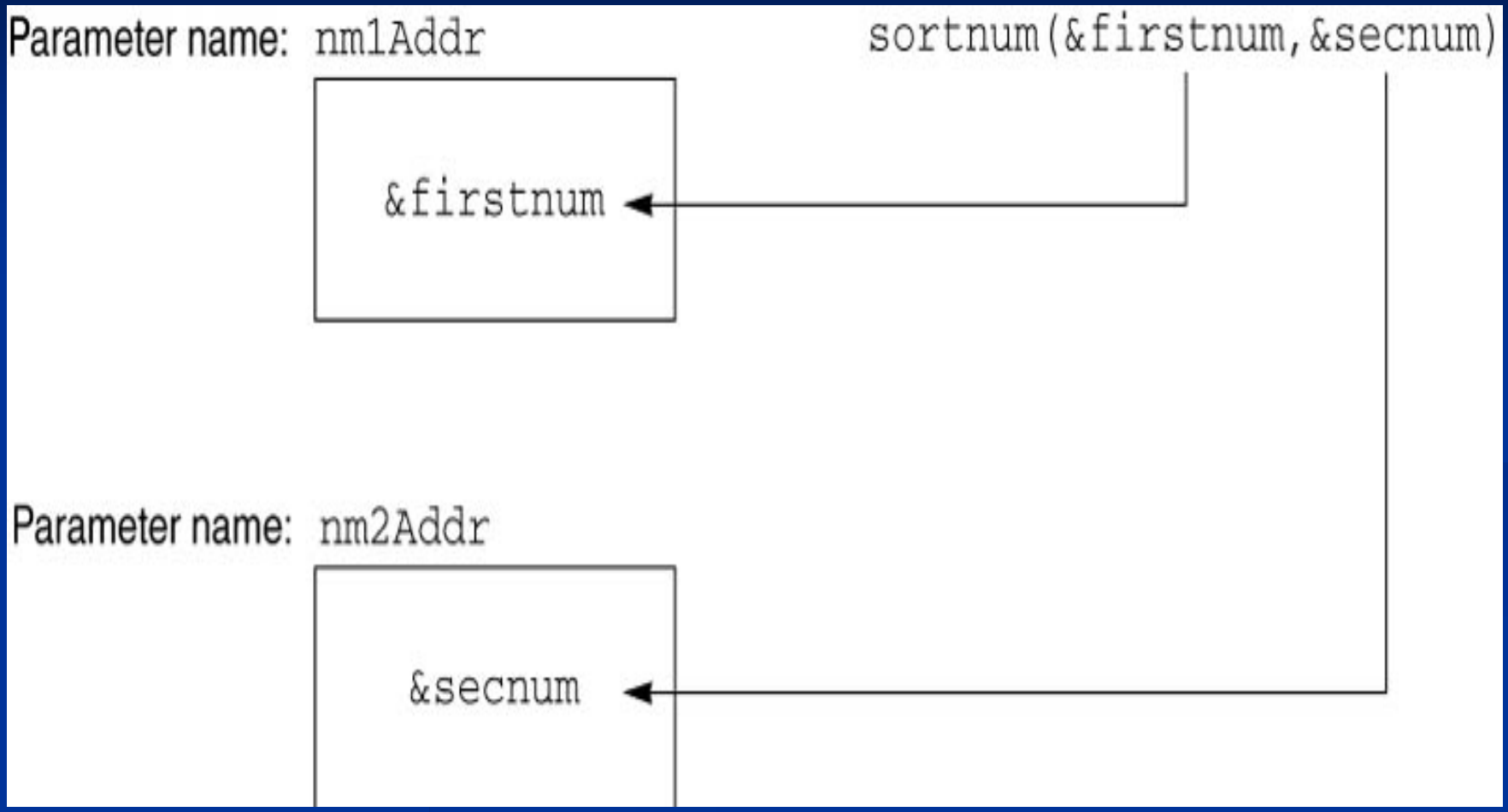
```
include <stdio.h>
void  displayChars ( char s,  unsigned int qty )
{
     unsigned int  i;
    printf("\n");
    for (i=0,  i<qty,  ++i)
          printf("%c", s);
     printf("\n");
}
```

传值调用

```
#include <stdio.h>
void sortnum(dou
int main()
{   double firstnun
    printf("Enter t
    scanf("%lf %l
    sortnum(&first
    printf("The smaller number is %6.2f", firstnum);
    printf("\nThe larger number is %6.2f\n", secnum);
    return 0;}
void sortnum(double *nm1Addr, double *nm2Addr)
{   double temp;
    if  (*nm1Addr > *nm2Addr)
      {    temp = *nm1Addr;        /* save firstnum's value */
          *nm1Addr = *nm2Addr;
           *nm2Addr = temp;        /* change secnum's value */
       }
    return;   }
```

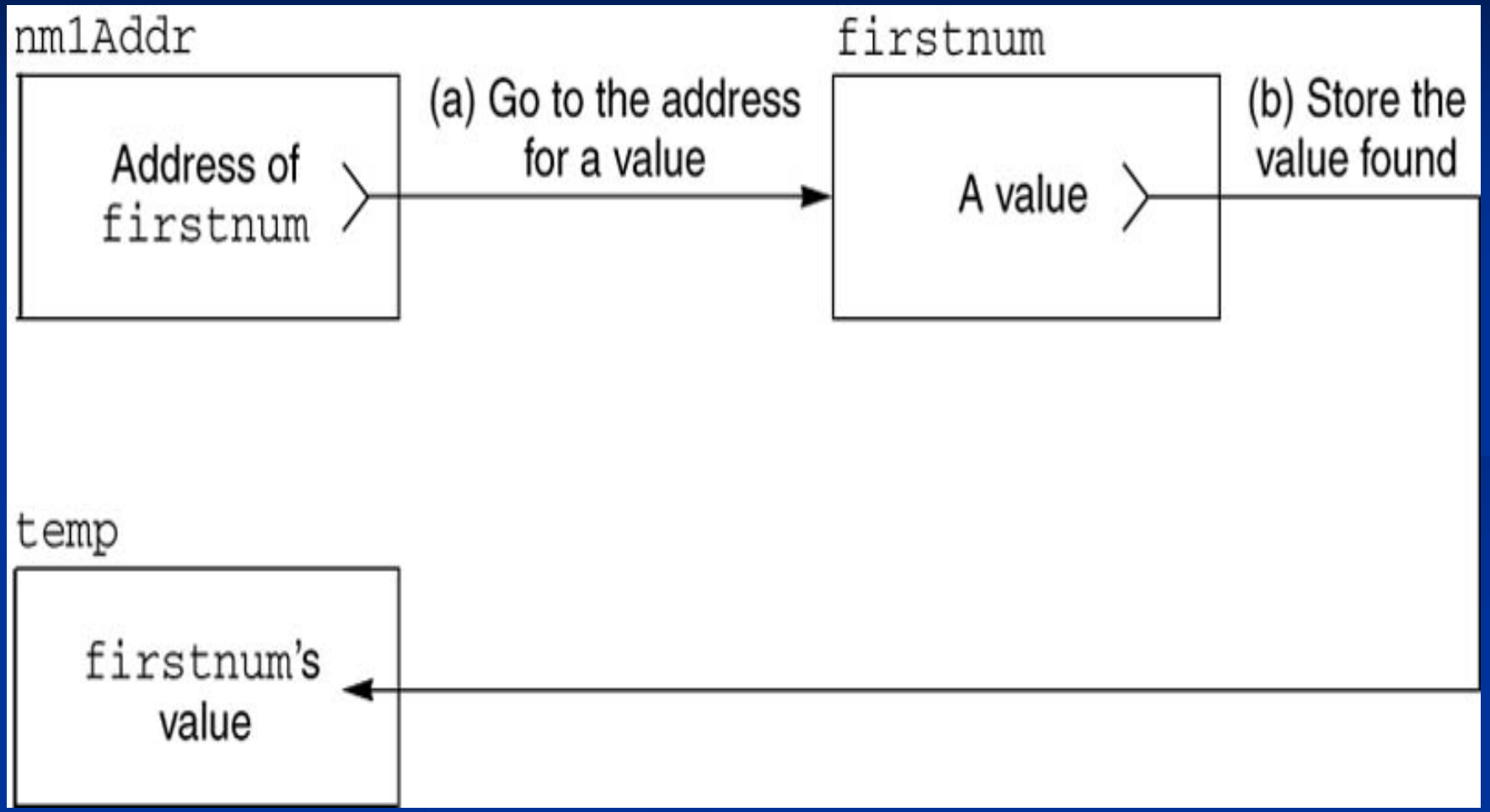


函数的传址调用



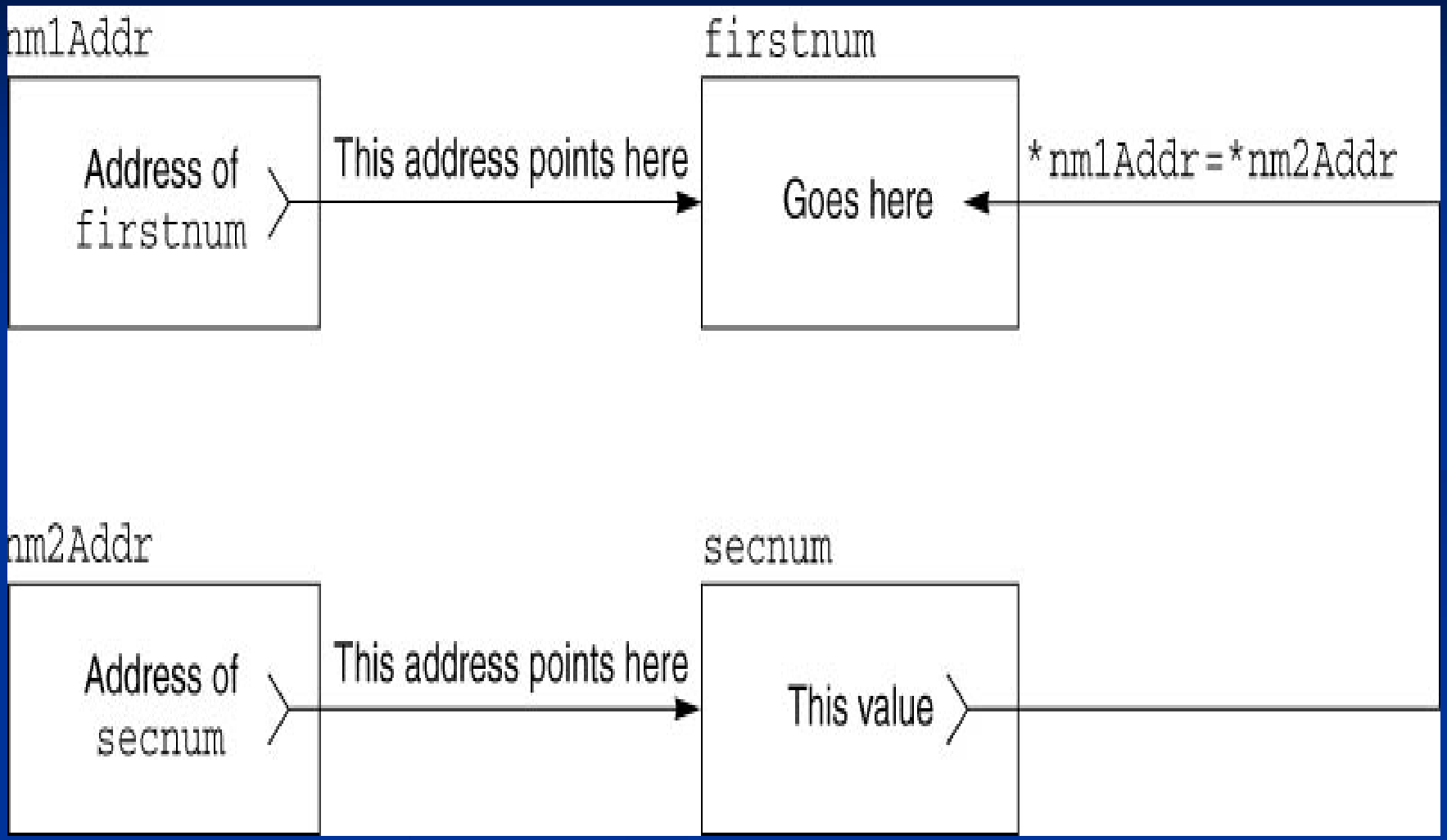
Storing addresses in parameters

函数的传址调用



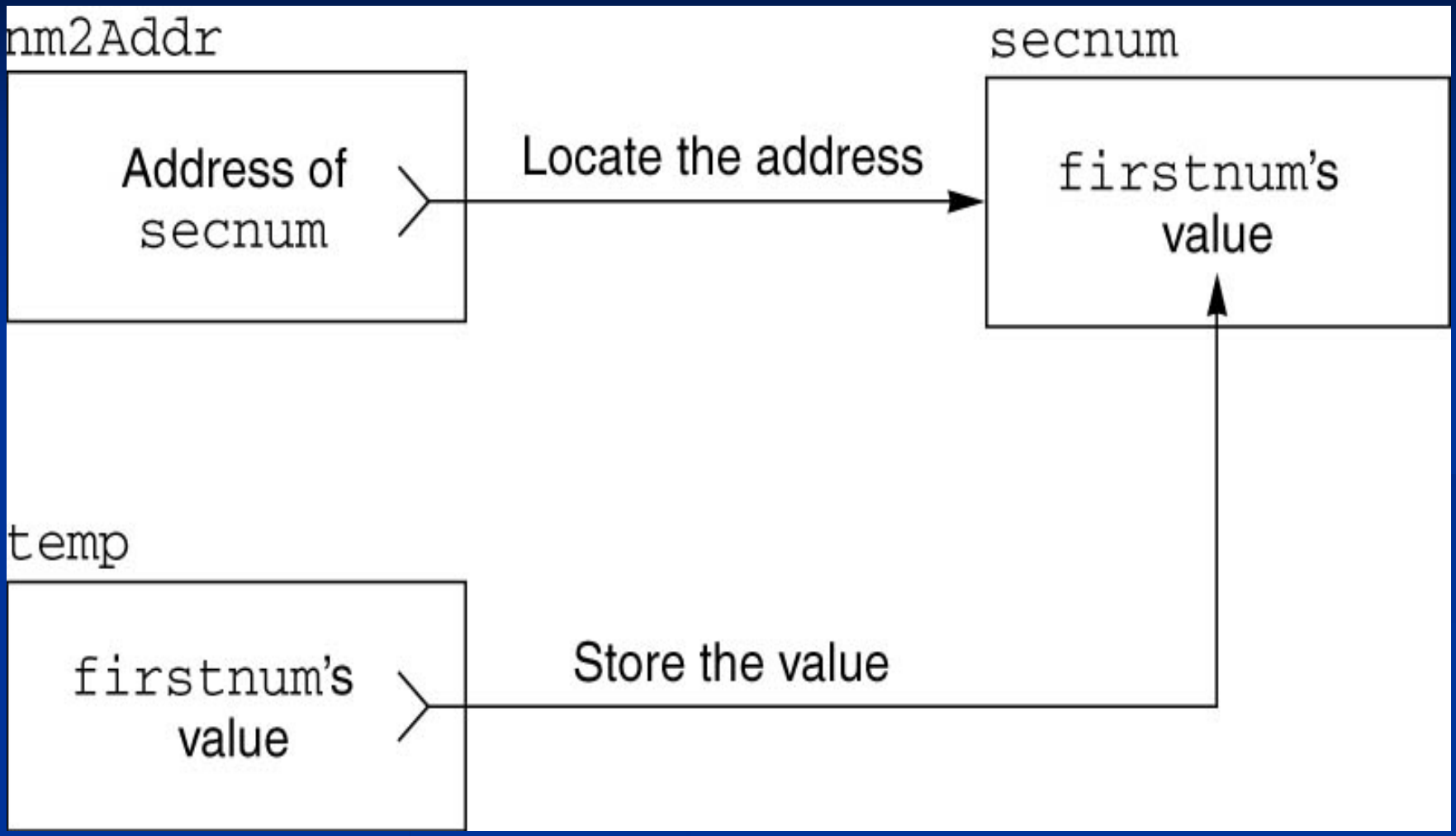
Indirectly storing *firstnum*'s value

函数的传址调用



Indirectly changing *firstnum*'s value

函数的传址调用



Indirectly changing secnum's value

C程序语句的排列:

```
#include  预处理说明
#define   符号常量名      表达式
函数原形

int main( )
{
        变量定义语句;
        其他语句;
        return  0;
}

函数定义
```

回顾

C程序语句的排列：

```
#include  预处理说明
#define   符号常量名      表达式
函数原形

int main( )
{
        变量定义语句;
        其他语句;
        return  0;
}

函数定义
```

回顾

- 函数定义的一般形式： function definition

返回值数据类型 函数名 （数据类型 形参1，数据类型 形参2 ，…，数据类型 形参n）

{

变量声明，如有变量;

其他语句;

return （ 表达式 ）;

}

void

回顾

- 函数原形的一般形式： function prototype
  - *返回值数据类型 函数名 （数据类型 参数1，数据类型 参数2，…，数据类型 参数n）；*
  - *返回值数据类型 函数名 （数据类型，数据类型，…，数据类型 ）；*
- 函数原形中的参数列表必需与函数定义中的形参列表保持一致：个数、顺序及各参数数据类型。

回顾

函数调用的三种方式：

按照函数在程序中出现的位置来分；

- 作为表达式出现在任何允许表达式出现的地方，参与运算。

  如：a＝sqrt(b)；

- 作为一条独立的语句完成特定的操作。

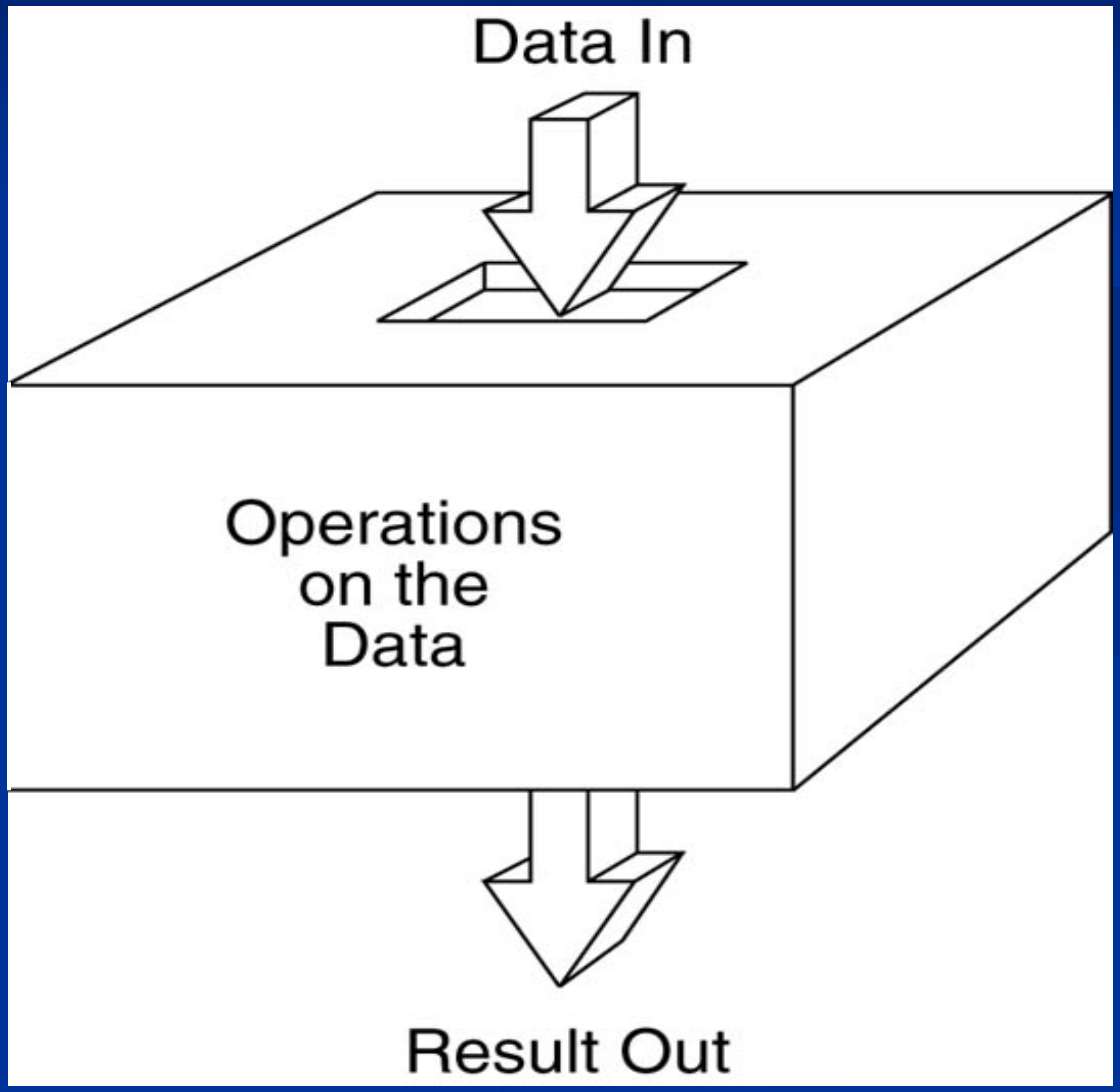
  如：gets(string1)；

- 作为函数的参数被其他函数调用。

  如： h=sqrt(abs(theta))；

编程的基本知识

—编程

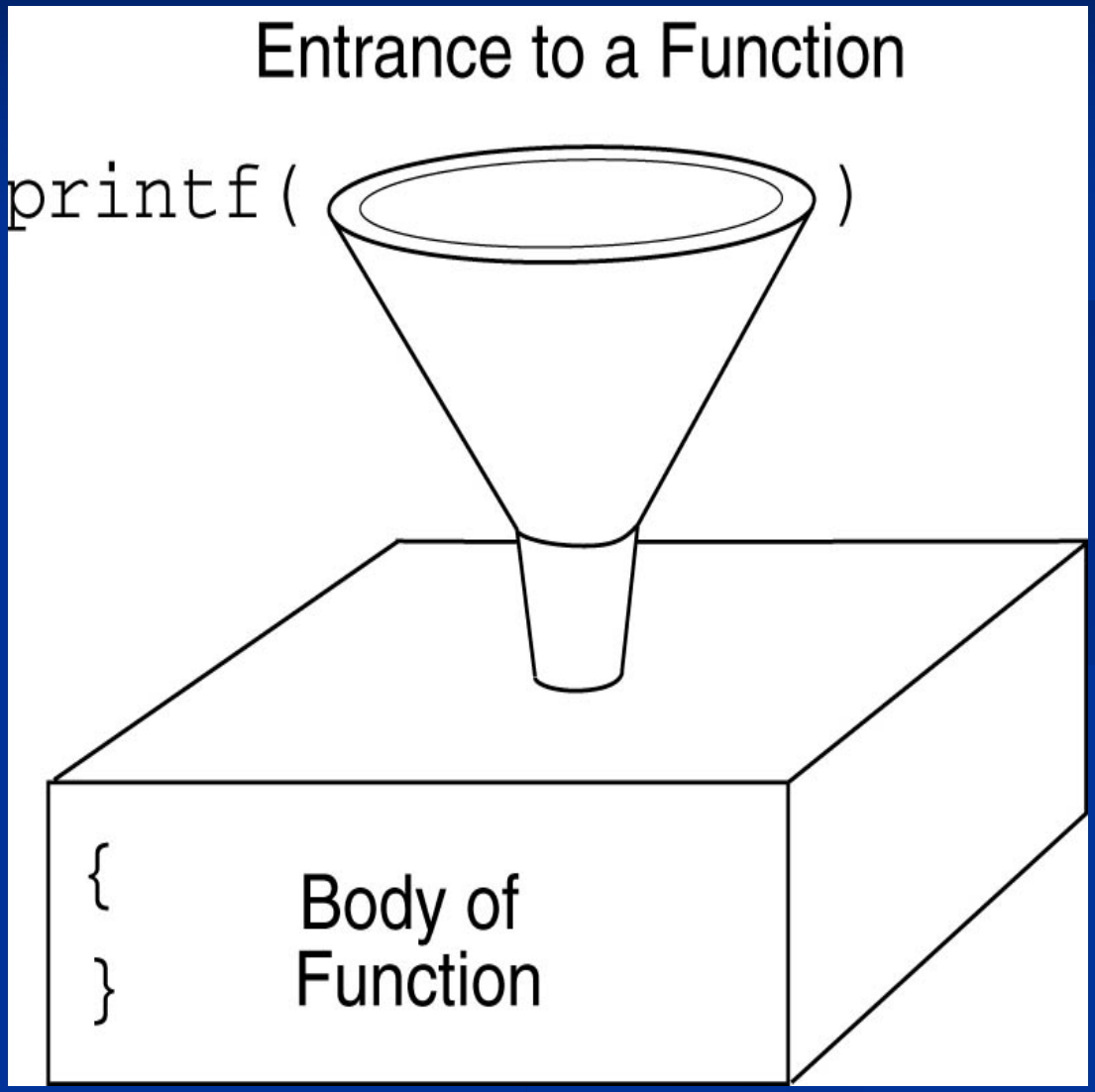


任一模块所完成的功能

在C语言中，使用函数来实现模块化。

编程的基本知识

—编程

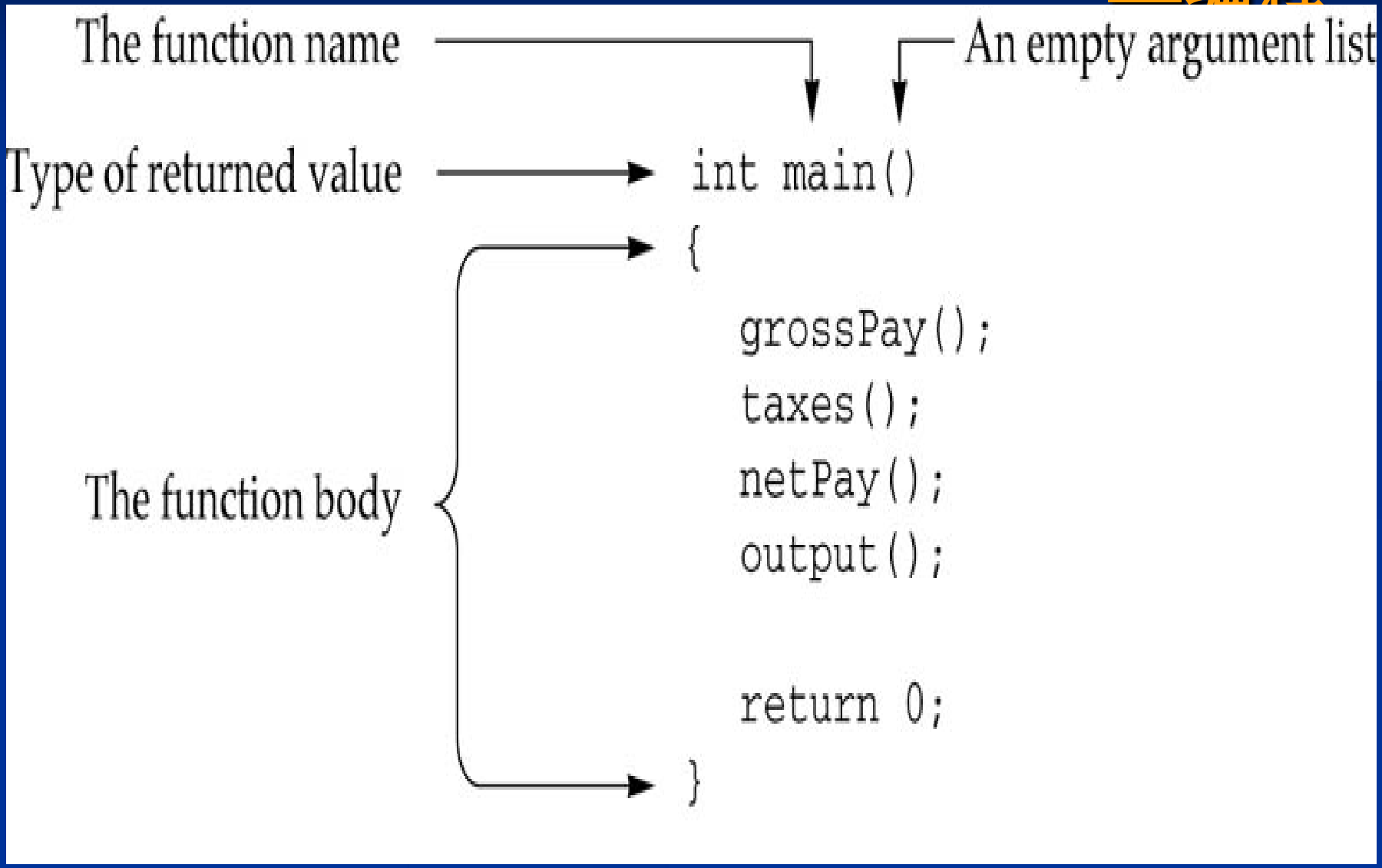


- 向 *printf( )* 传递信息

- *printf( )*是一种标准输出函数，由C编译系统的系统函数库提供。

编程的基本知识

——编程


The function name
An empty argument list
Type of returned value
int main()
{
grossPay();
taxes();
netPay();
output();
return 0;
}
The function body

回顾

- 强制类型转换运算符：　单目、2、右向左

（类型）表达式

float x=5.8, y=10.3

- (int) x + y
- (int) (x + y)

- 变量的作用域
- 变量的生命期
- 函数的传址调用

变量的作用域

- 变量的作用域 ：在变量占用存储空间的时间内是否能够被引用，即变量作用的有效范围。

  - *局部变量/内部变量*
  - *全局变量/外部变量*

变量的作用域

- **局部变量** 在一个函数内部定义的变量(内部变量)，它只在此函数范围内有效，在此函数以外不能被使用。
- **局部变量的有效范围：** 开始直到该函数结束。

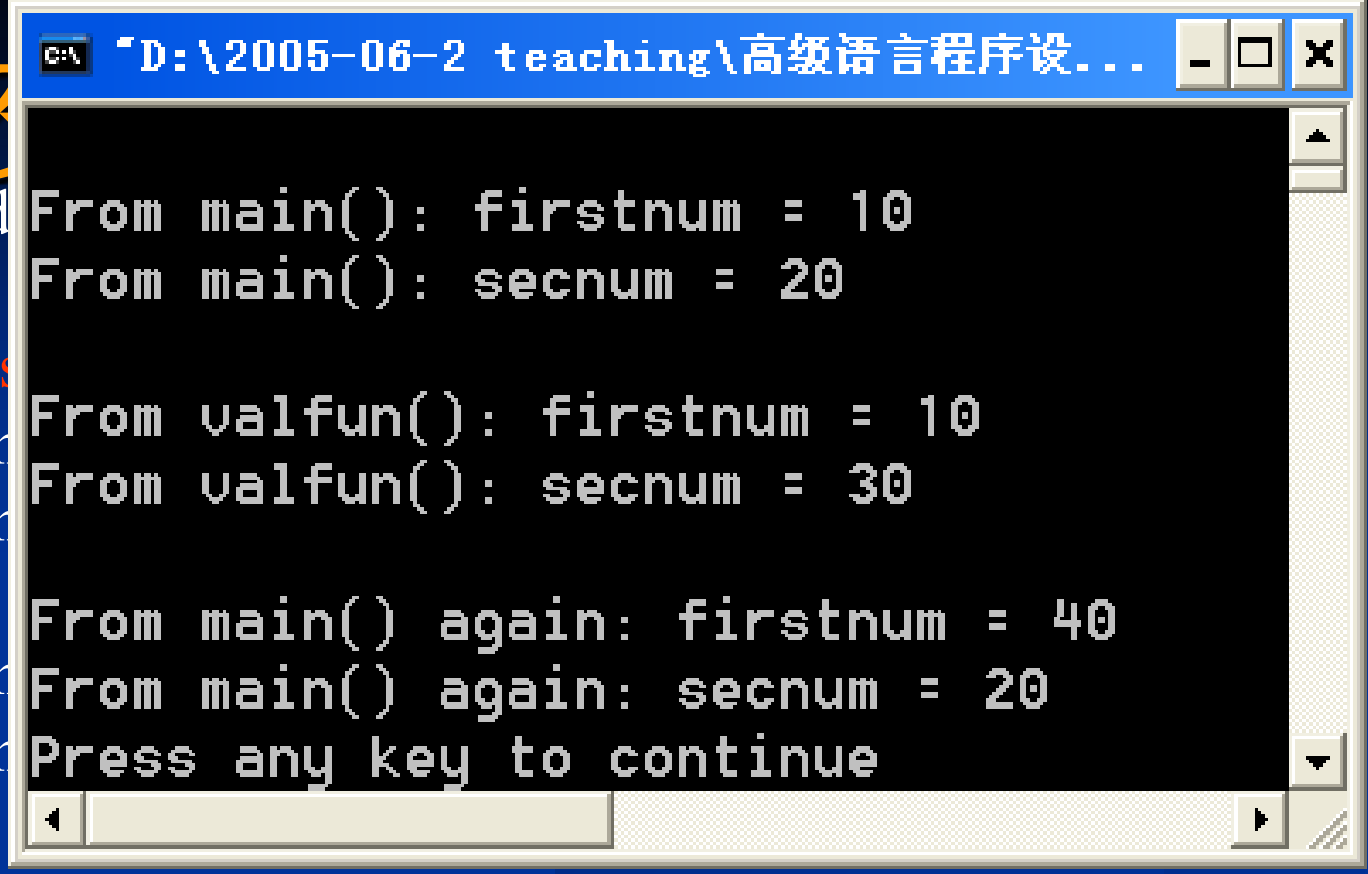
不同函数中使用同名变量？

- 形参也是局部变量。
- 在一个函数内部，可以在复合语句中定义变量，这些变量只在该复合语句中有效。这种复合语句称为“分程序”或“程序块”。

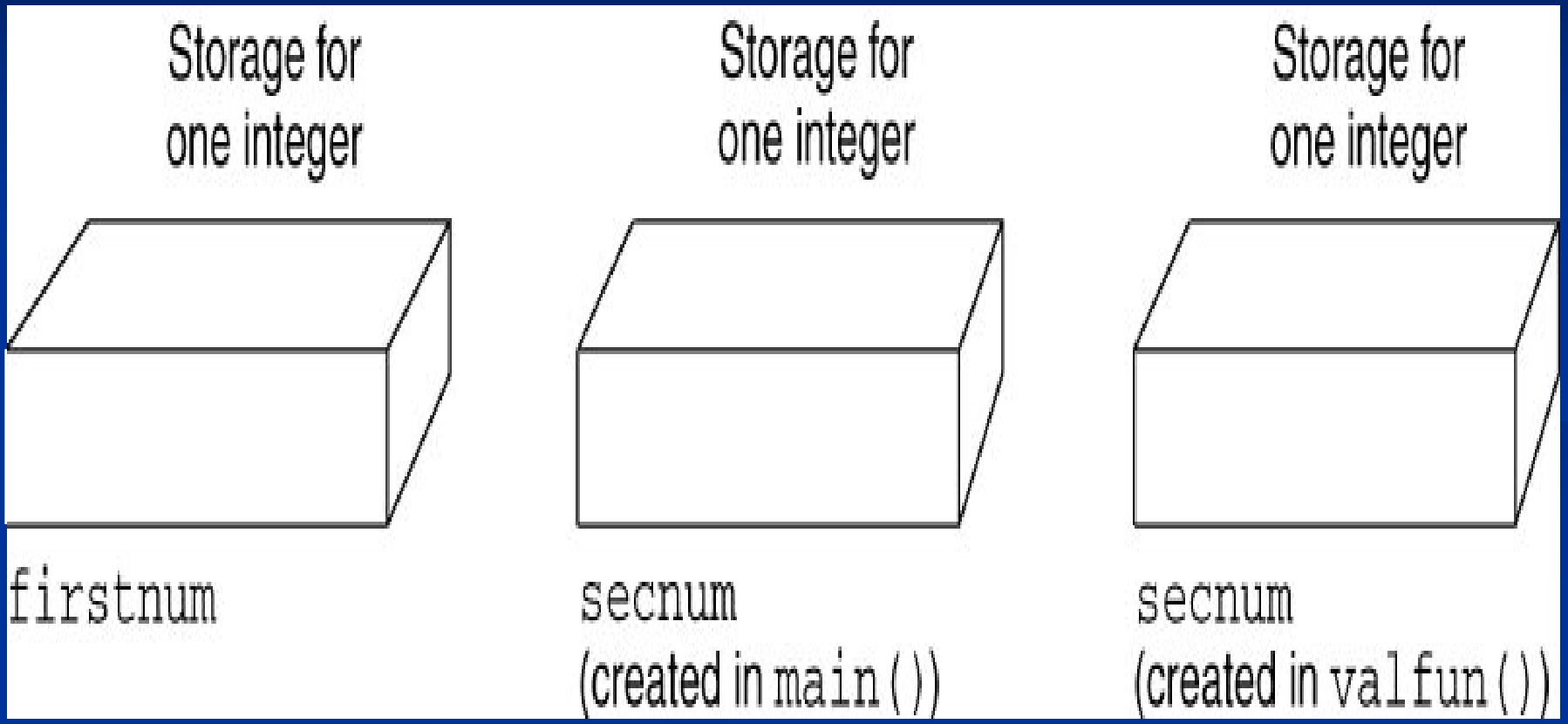
变量的作用域

- **全局变量** 在函数外部定义的变量(外部变量)。
- **全局变量的有效范围：** 从定义变量的位置开始直到本源文件结束。
- 设置全局变量的作用是增加函数间数据联系的渠道。
- 函数执行时若需要使用一个变量值，将首先查找局部变量区；找不到时，再到全局变量区查找。

```
#include <stdio.h>
int firstnum;       void
int main()
{  int secnum;       firs
  printf("\nFrom main
  printf("\nFrom main
  valfun();
  printf("\nFrom main
  printf("\nFrom main
  return 0;     }

void valfun()
{  int secnum;    secnum = 30;
  printf("\nFrom valfun(): firstnum = %d",firstnum);
  printf("\nFrom valfun(): secnum = %d\n",secnum);
  firstnum = 40;
  return; }
```

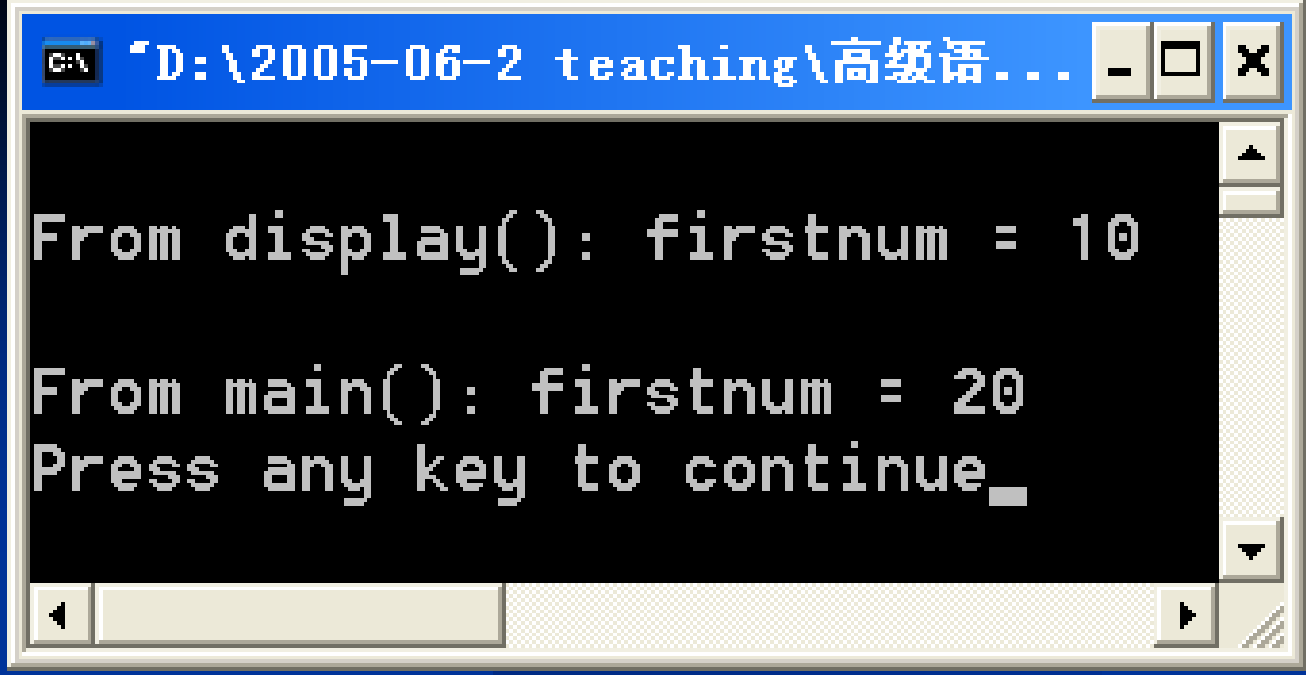


变量的作用域



The three storage areas created by the program

```
#include <stdio.h>

int firstnum=10;
void display();

int main()
{
  int firstnum = 20;
  display();
  printf("\nFrom main(): firstnum = %d\n",firstnum);

  return 0;
}

void display()
{
  printf("\nFrom display(): firstnum = %d\n",firstnum);
}
```



变量的作用域

在程序设计中，应尽量*避免使用*全局变量。

- 全局变量在程序的全部执行过程中都占用存储单元，而不是仅在需要时才开辟单元。
- 使函数的可靠性和通用性降低。
- 使用全局变量过多，会降低程序的清晰性。

变量的生命期

- 变量的生命期 ：变量占用存储空间的时限，由变量的存储类型决定。
- C语言中，变量的存储类型符有四种：
  - **auto**
  - **static**
  - **extern**
  - **register**
- C语言规定：存贮类型符放在变量定义的最前面。

例： **static int a, b;**

变量的生命期

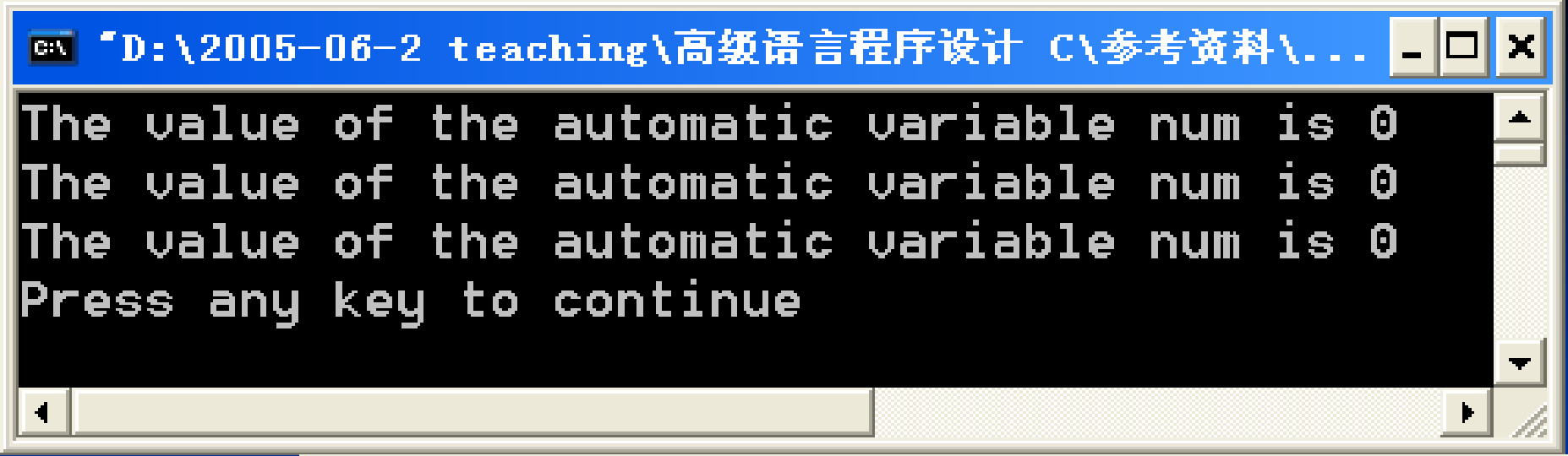
- 局部变量的存储类型只允许有三种：
  - auto
  - static
  - register

n 缺省时为auto。

例： auto int a，b；

int a， b；

- auto自动局部变量的生命期：在变量定义时获取存储空间（alive）；函数返回后，释放存储空间（die)。




```
  for(count =
    testauto( )
  return 0;

void testauto
{  static int n
  printf("The value of the automatic variable num is %d\n", num);
  ++num;
  return;    }
```