程序设计基础
(C Programming)

第二讲：C程序设计基础(二)

本章目标

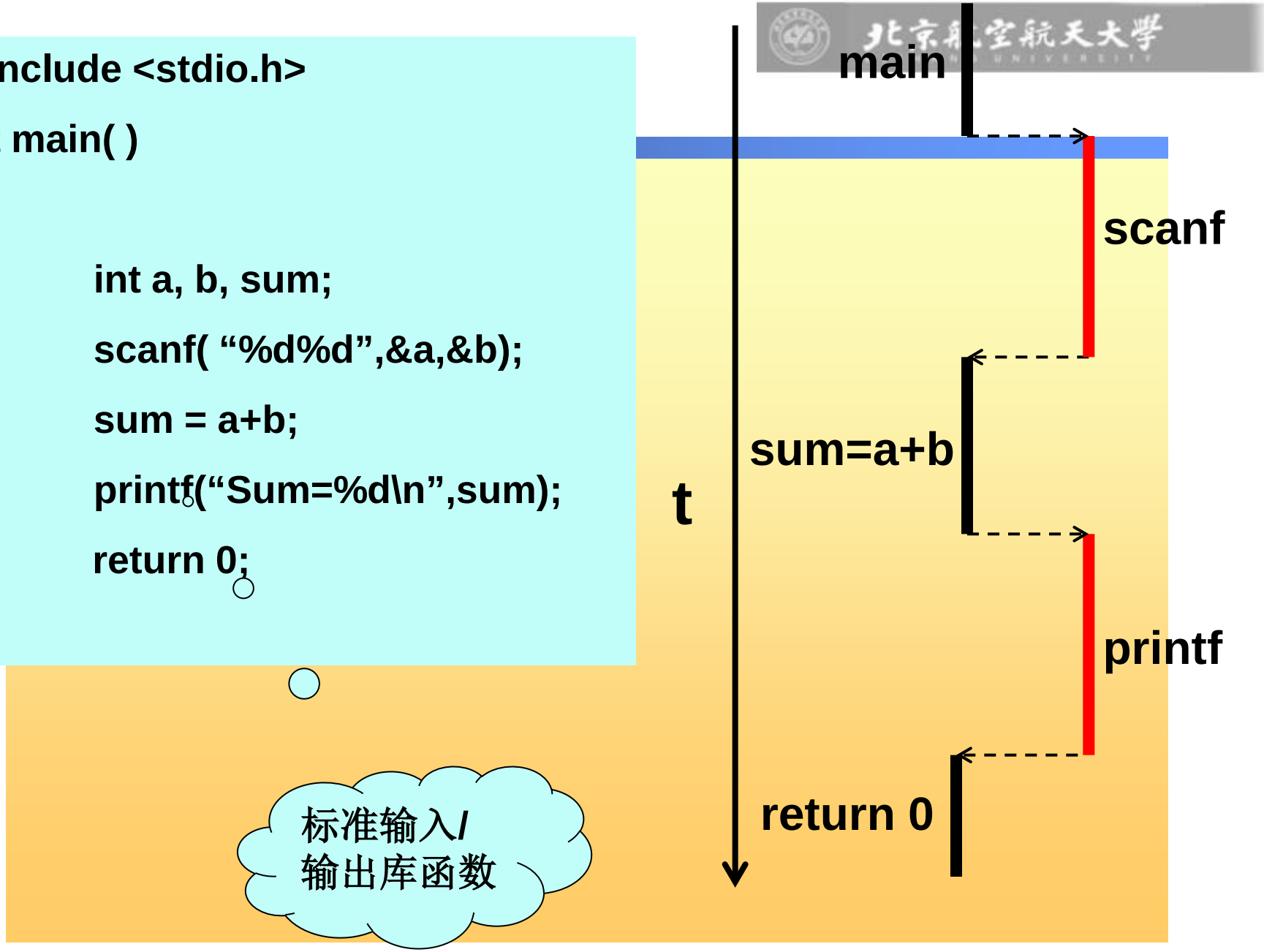
- 掌握函数(function)的定义及调用方式
- 掌握函数参数(parameter)传递方式
- 掌握一维数组(array)的定义和使用
- 掌握简单的文件(file)输入/输出

模块化程序设计

- **结构化程序设计**（structured programming）是最常用的程序设计方法之一，其特点为：
  - 自顶向下（top-down design）；
  - 逐步细化（stepwise refinement）；
  - **模块化**（modular programming）；
- **模块化的好处：**
  - 功能分解的需要（控制函数规模）；
  - 代码重用（减少重复代码）；

模块化是将复杂问题简单化的途径。

```
# include <stdio.h>
int main( )
{
        int a, b, sum;
        scanf( “%d%d”,&a,&b);
        sum = a+b;
        printf(“Sum=%d\n”,sum);
        return 0;
}
```



标准（库）函数

工欲善其事，必先利其器。

- 标准I/O库函数

  #include <stdio.h> **(scanf,printf, getchar, putchar…)**

- 标准数学函数

  #include <math.h> **(sin, cos, sqrt…)**

- …

常用标准库函数及头文件参见附录E。

问题2.1

- 问题: 已知一组三角形的三边（如(2.0, 2.0, 2.0)、(3.0,4.0,5.0)、(2.5, 3.1, 3.8)），计算其面积。计算三角形面积的公式为：

$$S=\sqrt{s(s-a)(s-b)(s-c)}, \text{其中} s=\frac{1}{2}(a+b+c)$$

```
#include <stdio.h>
#include <math.h>
int main ()
{
        double s,area;
        s = (2.0 + 2.0 + 2.0) / 2;
        area = sqrt(s*(s-2.0)*(s-2.0)*(s-2.0));
        printf("(2.0,2.0,2.0) area = %6.2f\n",area);
        s = (3.0 + 4.0 + 5.0) / 2;
        area = sqrt(s*(s-3.0)*(s-4.0)*(s-5.0));
        printf("(3.0,4.0,5.0) area = %6.2f\n", area);
        s = (2.5 + 3.1 + 3.8) / 2;
        area = sqrt(s*(s-2.5)*(s-3.1)*(s-3.8));
        printf("(2.5,3.1,3.8) area = %6.2f\n", area);
        return 0;
}
```

对于这样的有规律的重复计算，可以用更好的方法：**函数**来解决

double sqrt(double x)
为标准数学库中函数，
使用前应加上
#include <math.h>

定义求三角形面积函数

- 函数名：tri_area

•参数(parameter，又称为形参)：需要传递给函数的数据，包括参数名称和参数类型

double a, double b, double c　　三角形的三条边

- **函数计算结果（返回）类型**

double　　三角形面积

•函数是如何对数据进行计算(函数体）

```
double   tri_area(double a,double b,double c)
{
        计算  area = sqrt(s(s-a)(s-b)(s-c)), 其中s = 1/2(a+b+c)   语句

}
```

tri_area函数定义

函数（返回）类型

函数定义头部，其中tri_area为函数名double 为参数类型a,b,c为函数参数

```
double tri_area(double a, double b, double c)
{
    double s,area;
    s = (a + b + c) / 2.0;
    area = sqrt(s*(s-a)*(s-b)*(s-c));
    return (area);
}
```

局部变量

函数体

函数返回(return)语句,返回计算结果

函数定义与调用

- 在ANSI C标准中，函数定义形式为：

类型 函数名（参数说明）

{

[局部变量定义或说明]

语句

}

参数说明可以为空，多个参数以逗号分隔

函数定义与调用（续）

- 函数名一般是标识符，一个程序只有一个main函数，其它函数名可随意取，当然最好是有助于记忆的名字，如getchar函数。
- 在ANSI C标准中，函数（返回值）类型不允许省略，即使是返回整型值（int），当函数无返回值时，应其类型说明为void类型。
- 局部变量定义或说明可有可无。

若无返回值,函数用：
return;
或者函数中没有
return语句

注意：在C语言中，函数定义不允许嵌套，即在一个函数体内不能包含有其它函数的定义。

问题2.1：代码实现

函数原型

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

double tri_area(double a, double b, double c);

int main( )
{
    printf("(2.0,2.0,2.0) area = %6.2f\n", tri_area(2.0,2.0,2.0));
    printf("(3.0,4.0,5.0) area = %6.2f\n", tri_area(3.0,4.0,5.0));
    printf("(2.5,3.1,3.8) area = %6.2f\n", tri_area(2.5,3.1,3.8));
   return 0;
}
```

函数调用

```
double tri_area(double a, double b, double c)
{
        double s,area;
         s = (a + b + c) / 2.0;
        area = sqrt(s*(s-a)*(s-b)*(s-c));
        return (area);
}
```

函数定义与调用（续）

函数调用形式：*函数名（[实参表]）*

其中实参（argument）个数、类型、排列次序应和形参定义时一致。（老版本的C编译器往往不做这方面的检查）

函数通过return语句将值返回给调用函数。它有两种使用形式：

1）return 表达式;　　(函数有返回值)

2）return;　　(函数无返回值,对应函数类型为void)

注意：使用return语句只能返回一个值。

函数（返回值）类型说明

- 即函数定义时，函数返回值类型的说明，在ANSI C标准中，函数类型必须显式说明，当函数无返回值时（即类似于其它语言中的“过程”），则函数类型可说明成void。
- 在C语言中，函数返回值的类型可以是基本类型或指向其它类型的指针（也可返回**结构**或**联合**类型）。

函数原型说明（prototype）

- 在ANSI C标准中，所有函数必须要有原型说明，用以说明函数的返回值类型、函数参数类型、个数及次序。函数原型说明有两种形式：
  - 直接使用函数的头部（推荐使用）。如，

    double tri_area(double a, double b, double c);

  - 在原型说明中仅给出参数类型、个数及次序，无形参变量名。如， double tri_area(double , double, double);

函数原型类似于后面将介绍的外部变量说明，*在调用任何函数之前必须确保其已有函数原型说明或其函数定义在调用之前。*

注意：函数原型说明的类型、参数类型、个数及次序必须与函数定义时一致，否则会产生错误。

函数参数

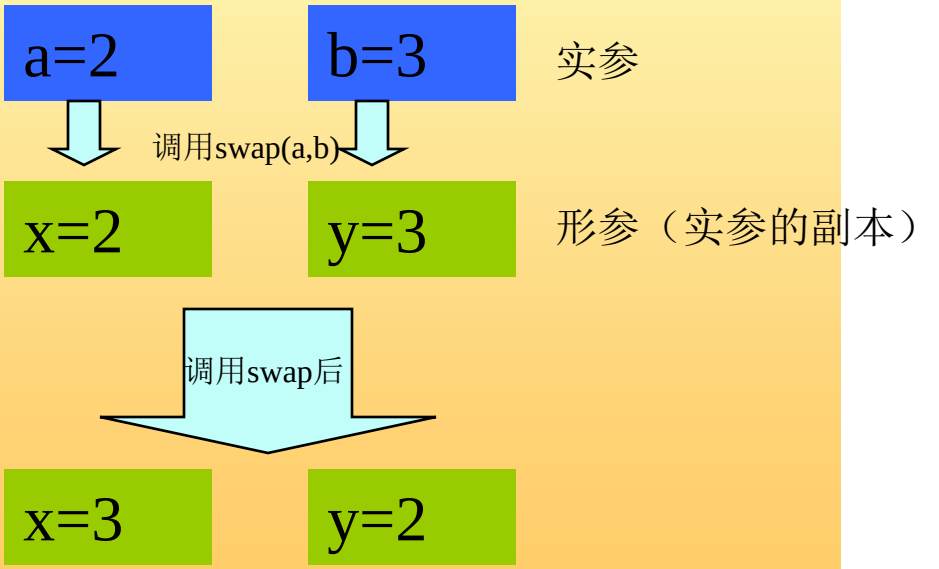
- 调用函数时，实参(argument)的类型、排列次序和个数应与函数定义时形参(parameter)相对应（在ANSI C标准中，若不一致，将出现编译错误）。
- C函数的参数传递全部采用**传值**(call by value)。传值调用实际上重新拷贝了一个副本给形参，因此，我们可以把函数形参看作是函数的局部变量。传值的好处是传值调用不会改变调用函数实参变量的内容，因此，可避免不必要的副作用。

函数参数（续）

- 如何理解“传值”调用？让我们来看一个试图交换两个数据值的例子。

```
void swap ( int x, int y)
{
        int temp;
         temp = x;
        x = y;
        y = temp;
}
int  main( )
{
        int a = 2,  b = 3;
        printf(“a=%d, b=%d\n”, a, b);
        swap(a, b);
        printf(“a=%d, b=%d\n”, a, b);
        return 0;
}
```

请问a和b的值是否交换？ 不能！



如何通过函数调用改变参数的值将在以后介绍。

函数参数（续）

由于函数调用时，参数传递是‘**传值**’，因此，在函数定义中的形参可当局部变量使用。下面求幂（x^n）函数power的实现是等价的：

```
double power(double x, int n)
{
        double p;
        for(p = 1; n > 0; --n)
                p *= x;
        return (p);
}
```

```
double power(double x, int n)
{
    int i;
    double p=1;
    for(i=1; i<=n; ++i)
        p = p*x;
    return (p);
}
```

问题2.2：求素数

- 问题2.2：读入一个整数，求该整数范围内的所有素数。
- 解题步骤：

```
读入一个整数到n;
for(i=2; i<=n; i++)
    if(i是素数)
        输出i;
```

可定义一个函数 int isprime(int n) 来判断一个整数是否为素数，若是返回1，否则返回0。

问题2.2：代码实现

```
int isprime(int n)
{
    int i;
    if(n<=1)
        return 0;
    for( i=2; i*i <= n; i++)
        if(n%i == 0)   /*存在因子，不是素数*/
            return 0;
    return 1;
}
```

主函数

```
#include <stdio.h>
int isprime(int n);
int main()
{
    int n,i;
    scanf("%d",&n);
    for(i=2; i<=n; i++)
        if(isprime(i))
            printf("%d ",i);
    return 0;
}
```

局部变量

注意：使用未初始化变量是程序员最常犯的错误之一。
提醒：在使用任何变量之前，确认其已有值。

- **局部变量（local variable）,又称自动变量(auto)：在函数（或块结构）中定义的变量。**
  - 只在定义它的**函数或块结构内有效**（即其使用范围，称为作用域scope）。（不同作用域可以有同名变量）
  - 编译程序不对局部（自动）变量给予隐含的初值，故其初值不确定。因此，**每次使用前，必须明确地置初值**。
  - 局部（自动）变量随函数的调用而存在，函数返回后将消失(生命周期lifecycle)，**由一次调用到下一次调用之间不保持值，每次调用函数时都重新初始化**。（动态分配）
  - 形参是自动变量，使用范围仅限于相应函数内。

不同作用域的变量可以同名，如：

```
int main()
{
    int i;
    …
    fun();
    …
}
void fun()
{
    int i;
    …
}
```

```
int main() /*读入10个字符 */
{
    int n,c;
    while( n++ < 10){
        c = getchar();
        …;
    }
    return 0;
}
```

错！

如何划分函数

- 程序中可能有重复出现的相同或相似的计算片段,可以考虑从中抽取出共同的部分,定义为函数。这样可以缩短程序代码，提高程序的可读性和易修改性。如例子2.1中的area()函数。（减少代码重复）
- 程序中具有逻辑独立的片段定义为函数。这样做主要用于分解程序，降低程序的复杂性。如例子2.2中的isprime()函数。（模块化）

问题2.3

- 问题：“从标准输入中输入10个整数，然后反序输出”。
- 输入样例：

  2332 455 5 67 78 9 10 467 3 23

- 输出样例：

  23 3 467 10 9 78 67 5 455 2332

问题2.3：问题分析

- 首先遇到的问题是如何保存输入的数据（为何要存所有输入数据）？
- 以目前所学的知识，我们可以设置10个变量来存储输入如：

```
int data1, data2, ..., data10;
```

使用数组！

- 这样做的缺点：
  - 程序处理数据非常烦琐，如我们必须依次读入每个数据（不能用循环）；
  - 程序不具扩展性，如果我们要处理100个、1000个甚至更多的数据，怎么办？
- 如何存储类型相同并且紧密相关的一组数据？

数组的定义与初始化

- 数组(array)是变量的有序集合，数组的所有成员（数组元素）都具有相同的类型。

- 数组定义一般采用如下格式：

  类型 数组名[长度]； /*长度为整型常量或常量表达式*/

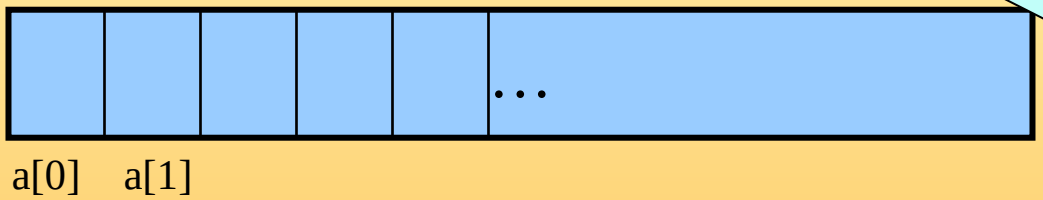
数组的定义与初始化（续）

数组元素的下标是从0开始，即数组中第一个元素的下标为0，最后一个元素下标为N-1（N为数组长度）

- 例如：

```
int  a[50];                    /*数组元素为a[0] ~ a[49] */

float m['c'-'a'], p[2*sizeof(double)];
```



sizeof是一个用来计算某个类型所占长度（以字节为单位）的运算符。sizeof(double)结果为8。

注意：对于一个长度为N的数组，其数据范围（下标）为0~N-1，而不是1~N(这是程序员常范的错误之一)。

数组的定义与初始化（续）

```
int length = 10;
double  s[length];
int  arry[ ];
```

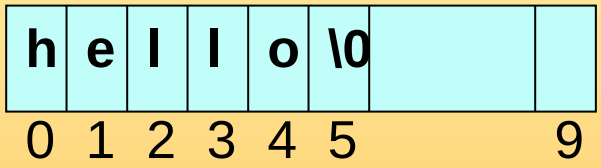
长度必须是常量
或常量表达式
也不能定义长度
为空的数组

注意：*C语言不支持动态数组，即数组的长度必须在编译时确定下来，而不是在运行中根据需要临时决定。但C语言提供了动态分配存贮函数，利用它可实现动态申请空间。*

数组的定义与初始化（续）

- 数组初始化：可以在定义时初始化一个数组。下面是一些数组初始化实例：

```
double sales[5] = {12.25, 32.50, 16.90, 23, 45.68};
double sales[ ] = {12.25, 32.50, 16.90, 23, 45.68};
int list[5] = { 6, 5, 12 }; //相当于：   int list[5] = {6,5,12,0,0};
int list[5] = {0};  //最常用的初始化数据组元素为0的方式
char string[10] = "hello";
char string[10] = {'h', 'e', 'l', 'l', 'o', '\0'};
char string[ ] = "hello";
```



注意：用字符串常量初始化一个字符数组时，数组长度应至少为字符个数多1。

数组元素的访问

- 初始化数组

```
int array[10], i;
for(i=0; i<10; i++)
    array[i] = 0;
```

或 **int array[10]={0};**

- 读入数组元素

```
int array[10], i;
for(i=0; i<10; i++)
    scanf("%d", &array[i]);
```

注意：不要将循环条件写为**i<=10**。这是初学者常犯的错误。如果一个数组的长度为**N**，则遍历一个数组的循环常写为：

for(i=0; i<N; i++)

或

for(i=0; i<=N-1; i++)

编译器不提供数组下标越界检查，运行时下标越界程序会出错

数组元素的访问（续）

- 数组中最大元素

```
maxIndex = 0;
for(i=0; i<N; i++)
    if(array[maxIndex] < array[i])
        maxIndex = i;
maxElement = array[maxIndex];
```

或：

```
max = array[0];
for(i=1; i<N; i++)
    if(max < array[i])
        max = array[i];
```

数组处理的限制

可以用下面库函数实现数据复制：

```
void *memcpy(void *dest,void *src, size_t count);
```

用法：

```
memcpy(y,x, sizeof(x));
```

- 在C中不允许对数组进行整体操作。下面用法是错误的：

```
int x[10], y[10];
scanf("%d", x);
y = x;
if( x == y )...
```

复制数组

比较数组

正确做法是：

```
for(i=0; i<10; i++)
    y[i] = x[i];
```

正确做法是：

```
for(i=0; i<10; i++)
    if(x[i] != y[i])
        …
```

问题2.3：解决步骤

1、定义一个长度为10的整型数组：int data[10]；

2、从标准输入中依次读入10个整数到数组data中；

3、从后往前输出数组data中每个元素；

问题2.3：代码实现：

```
# include <stdio.h>
#define NUM  10
int main( )
{
        int data[NUM], i;

        for(i =0; i <NUM; i++)
                scanf("%d", &data[i] );

        printf("\n");

        for(i =NUM-1; i >=0; i--)
                printf("%d ", data[i] );

        return 0;
}
```

依次读入**10**个整数到数组中。

从后往前输出数组中每个元素。

问题2.4

- **问题：**统计输入中每个小写字母的出现次数
- **方法分析：**
  - 方法一：显然可以使用26个int型变量来分别存储每个小写字母的出现次数。然后在程序中，使用switch或if_else if来分别统计每个小写字母的出现次数。（该方法太笨拙）

问题2.4：方法分析（续）

- 方法二
  - 可以使用数组来存储每个小写字母的出现次数。如： int lower[26];
  - 如何将输入的小写字母（字符型）与相应数组下标（整型）对应？

    表达式：

    c – ‘a’

    可将小写字母c转换为相应整数。因此，下面语句可累加每个小写字母的出现次数：

    lower[c-’a’]++;

问题2.4：代码实现

```
#include <stdio.h>
int main()
{
    int i, c, lower[26] = {0};
    while((c = getchar() ) != EOF)
        if(c >= ‘a’ && c <= ‘z’)
            lower[c -‘a’]++;
    for(i=0; i<26; i++)
        printf(“Number of %c: %d\n”, ‘a’+ i,  lower[i]);
    return 0;
}
```

问题2.4：思考*

- 本问题只统计小写字母的出现次数，如何统计字母出现次数（即大小写无关，'a'和'A'为同一字母）
- 如何统计字母（大小写分开）出现次数？
- 如何基于本问题解题思路，实现一个统计输入中字符的出现次数，并以直方图形式显示的程序？横直方图（较容易）、纵直方图（较难）？

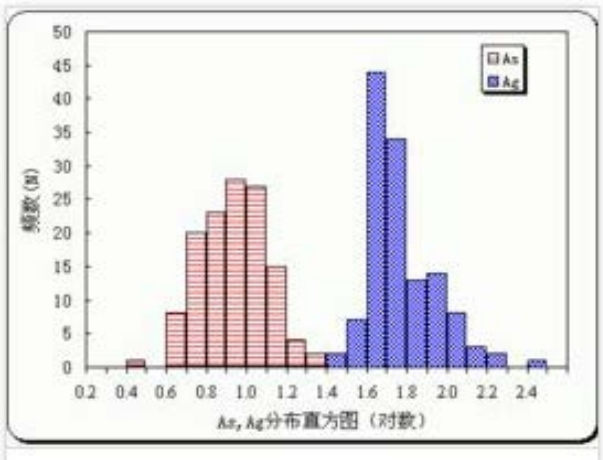
```
a:******************
b:********
c:**********************
d:**
…
```

横直方图

```
*
*     *
*   * *
**  * *
**  ***
*** ***
abcde….
```

纵直方图

```
/*统计字母出现次数*/
#include <stdio.h>
int main()
{
    int i, c, cc[128] = {0};
    while((c = getchar() ) != EOF)
        if(c >= 'a' && c <= 'z'|| c >= 'A' && c <= 'Z')
            cc[c ]++;
    for(i=0; i<128; i++)
        if(cc[i] !=0)
            printf("Number of %c: %d\n", i ,cc[i]);
    return 0;
}
```



数组作为函数参数

- **数组可以作为参数传递给函数。**实际上传递的是数组的首地址（即数组第一个元素的地址，将在指针部分说明），我们可以这样理解数组作为参数传递：形参数组与实参数组是一对共享同一数据区的数组，即它们是同一个数组，而不是对实参数组的拷贝。
- 非字符数组作为参数时，函数的定义形式：

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

- 数组作为参数时，函数的调用形式：

```
main()
{
   int a[10];
   ....
   fun(a, 10);
   ....
}
```

注意：定义数组形参时，数组长度可省略。对于非字符数组，还应在形参中指定数组元素个数。

注意：函数调用时，用数组名作实参。

数组作为函数参数（续）

```
// 例2.4 统计小写字母个数
#include <stdio.h>
int main()
{
    int i, c, lower[26] = {0};
    while((c = getchar() ) != EOF)
        if(c >= 'a' && c <= 'z')
            lower[c -'a']++;
    print(lower, 26);
    return 0;
}
```

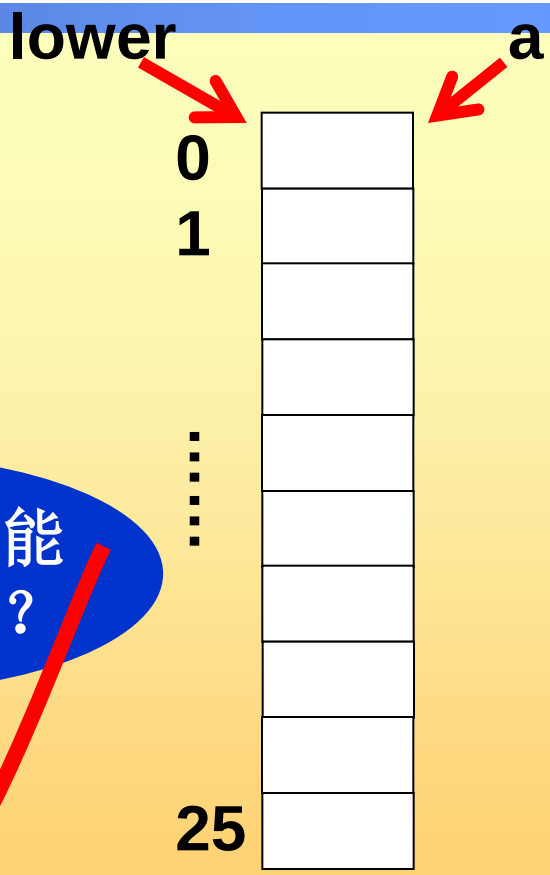
```
void print( int a[ ] , int length)
{
        int i;
        for(i=0; i<length; i++)
            printf(" %d\n", a[i]);
}
```

注意：数组长度不要写一维数组形参内，它无法传递给函数。

数组作为实参，千万不要写成 lower[26]

数组作为函数参数（续）

```
void  count( int a[ ] )
{
        int c;
        while((c = getchar() ) != EOF)
                if(c >= 'a' && c <= 'z')
                        a[c -'a']++;
}
int main()
{
        int lower[26] = {0};
        count(lower);
        print(lower, 26);
        return 0;
}
```



数组中的值能传递回来吗？

字符数组

数组元素的类型是char。

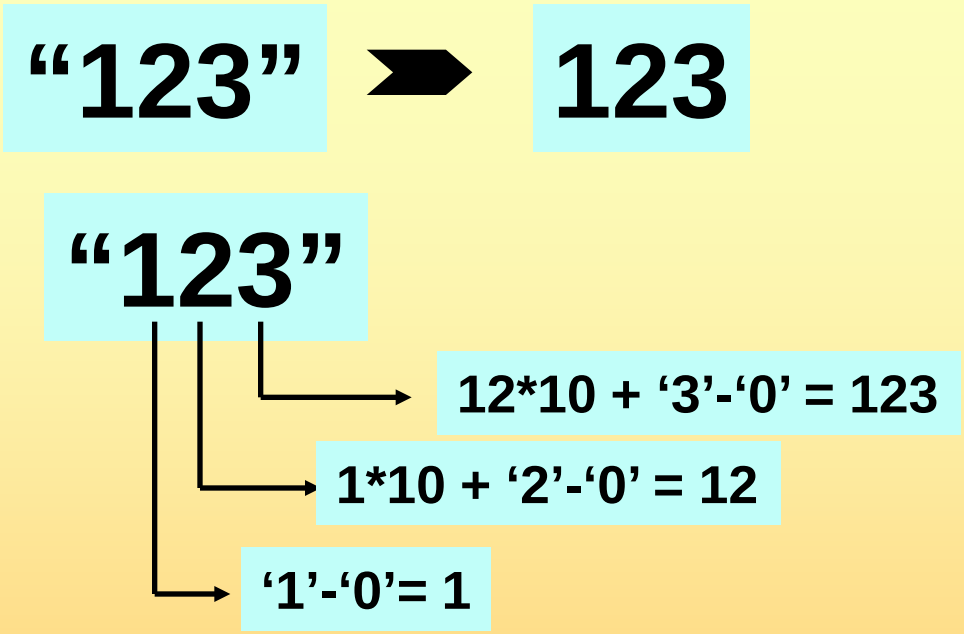
char mes[ ] = “C Language”;

char line[100] = “Programming”;

字符数组有如下特点：

- 数组元素跟一般变量一样可赋值、比较、计算等。
- 数组下标也是从0～N-1（N为数组长度）。
- 字符数组长度可以显式给出，也可以隐式得到。
- 由双引号括起来的的字符串常量具有静态字符串数组类型。
- 用字符串常量对数组初始化时，编译程序以\0作为结束这个数组。因此，**用字符数组来存放字符串时，数组长度要比字符串长度多1**。

问题2.5：将数字字符串转换成整数



方法分析

```
n = 0;
for(i=0; s[i]为数字字符; i++)
        n = 10*n + s[i] - ‘0’;
```

问题2.5：代码实现

```
int atoi(char s[ ])
{
    int i, n, sign;
    for(i=0; s[i] = = ' ' || s[i] = = '\n' || s[i] = = '\t'; i++)
        ;                /* skip white space */
    sign = 1;
    if(s[i] = = '+' || s[i] = = '-')
        sign = (s[i++] = = '+')?1:-1;
    for(n=0; s[i] >= '0' && s[i] <= '9'; i++)
        n = 10*n + s[i] – '0';
    return ( sign * n);
}
```

字符数组。

空语句

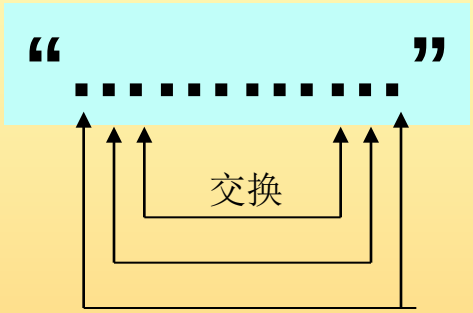
条件运算符：
<表达式1> ? <表达式2> : <表达式3>
先计算表达式1，若其值为非零，则整个表达式结果为表达式2的值，否则就为表达式3的值。

```
#include <stdio.h>
int atoi(char s[ ]);
int main()
{
    char s[20];
    scanf("%s", s);
    printf("%d\n", atoi(s));
    return 0;
}
```

读入一个以空白字符分隔的字符串到s中，以'\0'结束(即由非空字符组成的字符串)。

为何字符数组（字符串）作为函数参数传递时，通常不用传递数组长度？

问题2.5：将字符串颠倒



方法分析

字符数组作为函数参数传递时，不需要同时传递数组长度。因为字符数组中字符串是以'\0'结束的。

当有多个循环变量时，要用逗号隔开。
逗号表达式：如*e1,e2*顺序求e1和e2，以e2值作为整个表达式结果的值。如，
a = (t = 3, t+2);
结果为**5**

```
void  reverse(char s[ ])
{
    int c, i, j;
    for(i=0, j=strlen(s)-1; i < j ; i++,j--){
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
}
```

```
int strlen(char s[ ])
{
    int i=0;
    while(s[i] != '\0') ++i;
    return (i);
}
```

```
#include <stdio.h>
void reverse(char s[ ]);
int strlen(char s[ ]);
int main()
{
    char s[20];
    scanf("%s", s);
    reverse(s);
    printf("%s\n",s);
    return 0;
}
```

输出一个以'\0'结束的字符串。

提示：数组作为函数参数传递时，要不要同时传递数组长度，取决于函数中是否知道数组中元素的个数。

简单文件操作

- 到目前为止，所有输入/输出操作均为标准输入/输出（即针对键盘及屏幕）。
- 实际应用多数都是针对文件输入/输出，如Office应用。

问题2.7：文件拷贝

- 问题： 将一个给定文件“input.txt”中内容拷贝到文件“output.txt”中。

将标准输入拷贝到标准输出程序非常简单：

```
#include <stdio.h>
int main()
{
    int c;
    while((c=getchar()) != EOF)
        putchar(c);
    return 0;
}
```

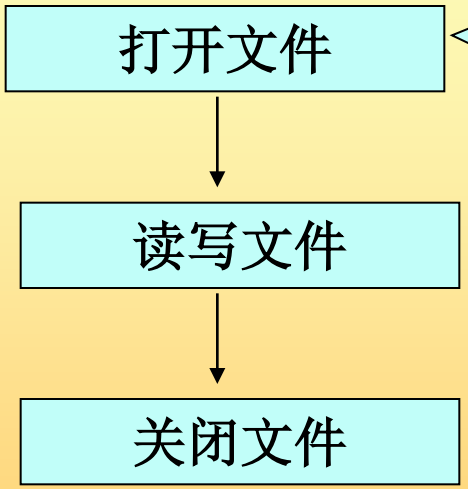
如何对文件读写？

文件输入/输出

首先在程序文件的头部应有如下语句：

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

- 文件输入/输出过程




```
FILE *in, *out;

in = fopen("input.txt", "r"); //为输入打开一个给定文件“input.txt”；打开方式”r”为以只读方式打开一个文件。
out = fopen("output.txt","w"); //为输出打开一个给定文件“output.txt”；打开方式”w”为打开一个只写文件。
```

```
c =fgetc(in);  //从文件（input.txt）中读入一个字符

fputc（c,out); //输出一个字符到文件（output.txt）中
fgets(s, n, in); //从 文件in上读入一行（最多读入n-1个字符），放入s 字符数组中。返回s或NULL。
 fputs( s, out); //把字符串s（不一定含\n）写入文件out中。返回非负数或EOF
fscanf(in, "%d", &score); //从文件in中读入一个整数
fprintf( out, "%d\n", score); //输出一个整数到文件out中
```

```
fclose(in); // 关闭文件input.txt
fclose(out); //关闭文件output.txt
```

问题2.7：代码实现

```
#include <stdio.h>
int main()
{
    int c;
    FILE *in, *out;

    in = fopen("input.txt", "r");
    out = fopen("output.txt","w");
    while((c= fgetc(in) ) != EOF)
      fputc(c, out);

    fclose(in);
    fclose(out);
    return 0;
}
```

为读写文件定义文件指针

打开文件
文件input.txt和output.txt位于与该执行程序.exe文件同一目录下（在VC中则与工程在同一目录下）

从文件**input.txt**中依次读一个字符

向文件**output.txt**中依次写一个字符

关闭两个打开的文件

问题2.7：测试

- 首先在该程序工程文件（即.dsp文件）目录下准备一个包含一定内容的input.txt文件。
- 程序正确运行后，将在同一目录下生成一个output.txt文件。
- 检查文件input.txt和output.txt内容是否完全一样。

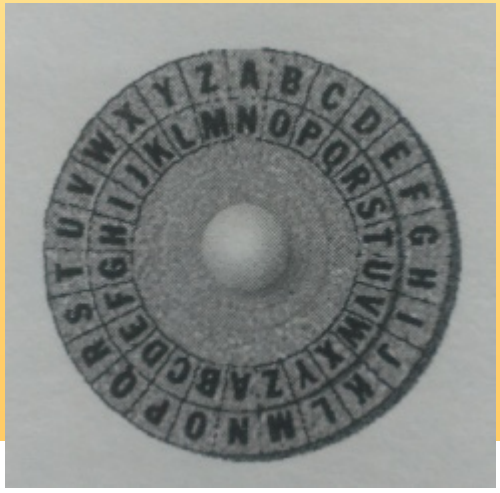
问题2.7：代码实现*

```
/* c2_7.c */
#include <stdio.h>
int main()
{
    int c;
    char infile[32],outfile[32];
    FILE *in, *out;
    printf("input file:");
    scanf("%s",infile);
    printf("output file:");
    scanf("%s",outfile);
    in = fopen(infile,"r");
    out = fopen(outfile,"w");
    while((c=  fgetc(in) ) != EOF)
        fputc(c,out);
    fclose(in);
    fclose(out);
    return 0;
}
```

问题2.8：凯撒（Julius Caesar）文件加密*

- **问题：** Julius Caesar（凯撒）加密方法：在每次加密时都选定一个加密密钥，它是一个1到25之间的数字，用于指定加密字母时的移位个数。例如，如果密钥为3，则将A转换为D，将Z转换为C，依次类推。小写字母亦如此，其它字符不变。例：若密钥为3，文本“C Language is Important.”，加密后为“F Odqjxdjh lv Lpsruwdqw.” 。

  从标准输入中读入一个要加密的文件名和加密后的文件名，用该方法实现对文件加密。



市面上名为“凯撒大帝”的玩具

问题2.8：凯撒（Julius Caesar）文件加密*

- 方法分析

  若c为一个小写字母：

‘a’ + (C - ‘a’ + key) % 26

问题2.8：凯撒（Julius Caesar）文件加密*

```
/* c2_8.c */
#include <stdio.h>
int main()
{
    int c, key;
    char infile[32],outfile[32];
    FILE *in, *out;
    scanf("%s %s %d",infile,outfile, &key);
    in = fopen(infile,"r");
    out = fopen(outfile,"w");
    while((c=  fgetc(in) ) != EOF)
        if(c>='a' && c<='z')
            fputc('a' + (c – 'a' + key)%26, out);
        else if( c>='A' && c<='Z')
            fputc('A' + (c – 'A' + key)%26, out);
        else
            fputc(c,out);
    fclose(in); fclose(out);
    return 0;
}
```

程序设计实践：调试程序(二)

- 如何调试函数？
- 如何查看字符数据？
- 如何查看数组中的数据？

函数调试方法：

1. 在该函数调用前设置一个断点
2. 当程序执行到该函数调用前断点时，查看每个函数实参值是否正确
3. 使用单步执行（Step Into)到该函数内
4. 查看每个参数是否被正确传递
5. 然后按照常规调试方法调试函数内语句

(以问题2.5颠倒字符串程序为例)

常见问题

- 读double类型数据错误,如:

```
double  a;
scanf("%f", &a);
```

正确用法:

```
double a;
scanf("%lf", &a);
```

- 使用未初始化的局部变量,如:

```
int  n;
while(n++ < 10)
    ...
```

正确用法:

```
int n = 0;
while(n++ < 10)
    ...
```

- scanf中空格使用错误，如:

```
int data1, data2;
char  op;
scanf("%d%d%c", &data1,&data2,&op);
```

正确用法:

```
int data1, data2;
char  op;
scanf("%d  %d  %c",
  &data1,&data2,&op);
```

常见错误

- 数组只能在定义时进行整体初始化赋值；但不能通过赋值运算符进行整体赋值。如下面初始化数组错误：

```
int  mon[13];
mon[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
```

正确用法：
```
int mon[13]={0,31,28,31,30,
31,30,31,31,30,31,30,31};
```

- 数组访问越界错误，如：

```
int  i,array[10];
for(i=1; i<=10; i++)
   scanf("%d",&array[i]);
```

正确用法：
```
int  i,array[10];
for(i=0; i<10; i++)
   scanf("%d",&array[i]);
```

- 数组作为函数参数传递错误
  - 函数调用时错误,如:

```
fun(a[10]);
```

正确用法：
```
fun(a);
```

- 打开（某绝对路径下）文件错误

```
in  = fopen("d:\input.txt", "r");
```

正确用法：
```
in = fopen("d:\\input.txt", "r');
```

本讲结束