计算机导论与 C 语言基础

计算机导论与 C 语言基础课堂笔记

Jinna

October 2016

第 1 章

C 语言中的字符串

1.1 字符串初始化

char a[3] = 'a','b','c';

char c[] = "China"; 双引号中的字符是字符串，字符串最后会默认一个“
0”，换言之，所有以“
0” 结尾的字符数组都可以被看成是字符串。这样系统分配给这个字符串分配一个比字符数大一的内存。所以，char c[5] = "China"; 是错的。

* 注意：字符数组只可以进行初始化，不能用赋值语句将一个字符串常量或字符数组直接赋给另一个字符数组。

1.2 字符输入

1.2.1 直接用 cin 输入字符

cin 从缓冲区读取了数据时，会将所有的空格和回车当作输入数据之间的间隔符，从输出可以看到，既没输出空格也没输出回车。(control + z) 是结束 cin 输入的标志。

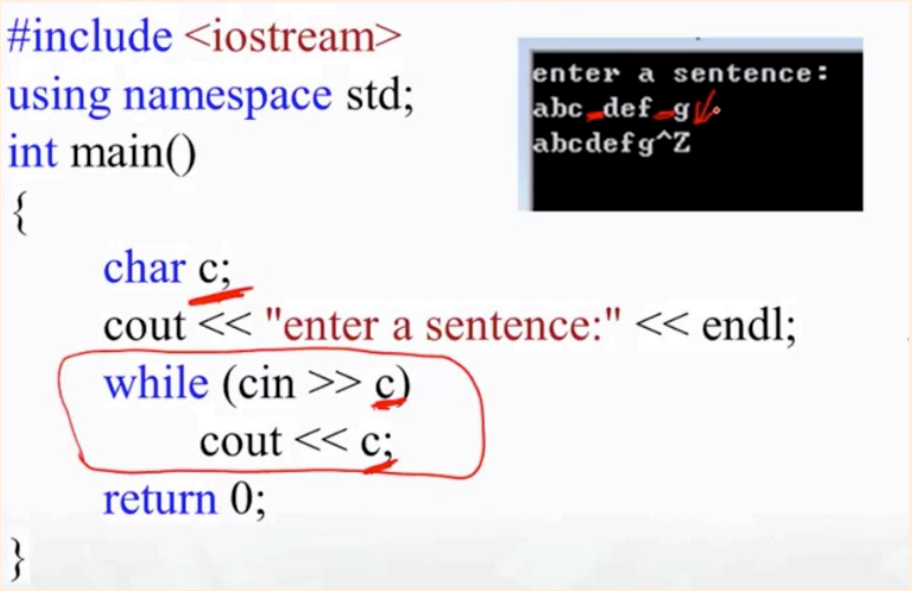


图 1: 直接用 cin 输入字符示例

1.2.2 用 cin.get() 函数输入字符

EOF：文件结束标志。cin.get() 可以将空格和回车都当作字符输入，并打印，不会跳过空格和回车。程序结束输入 control + z;

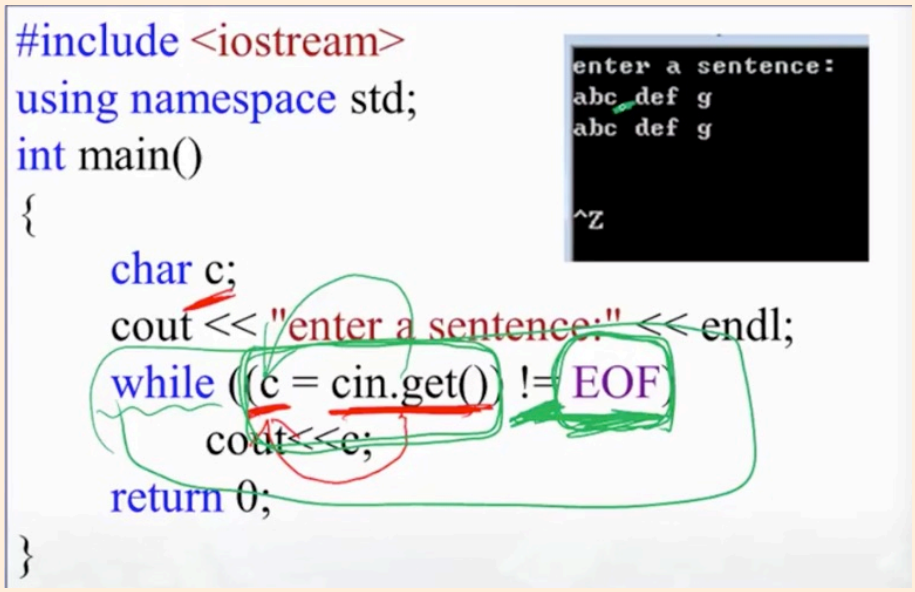


图 2: 用 cin.get() 函数输入字符示例

1.2.3 用 cin.get(char) 输入字符

定义带参数的函数。输出结果与不带参数的 cin.get() 一样。

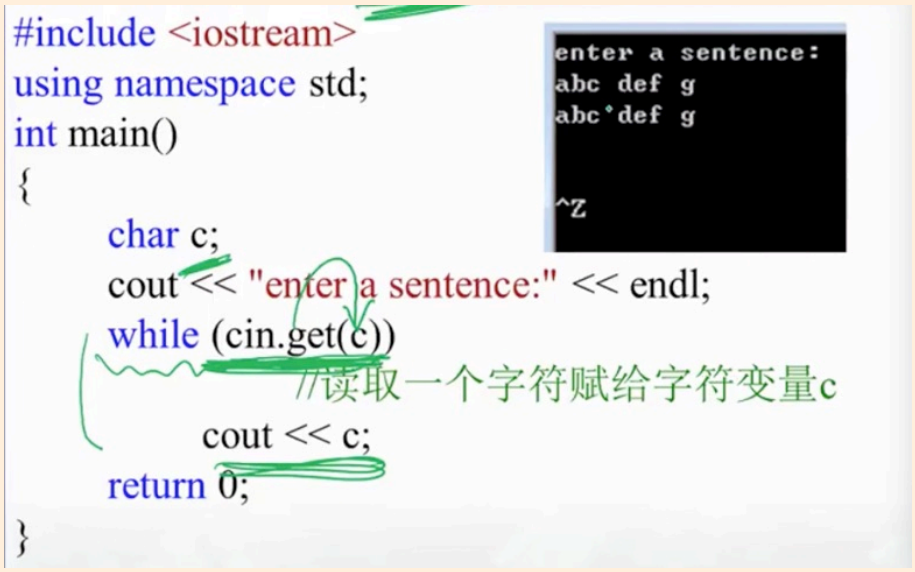


图 3: 用 cin.get(char) 输入字符示例

1.2.4 用 getchar() 输入字符

用 getchar() 输入，不会跳过任何字符。即使 control + z 也会被打印输出。

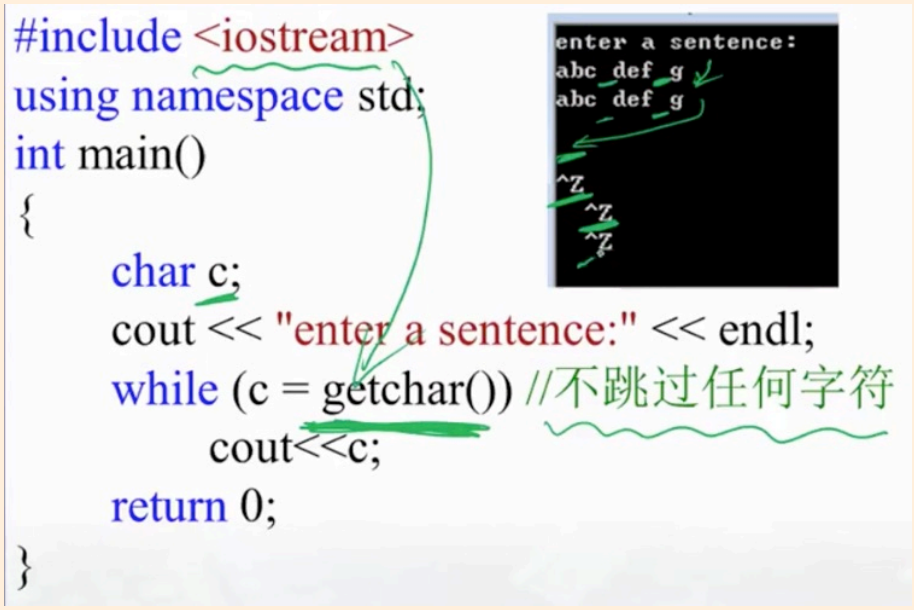


图 4: 用 getchar() 输入字符示例

1.3 字符输出

1.3.1 用 cout 输出字符

将定义的数组直接输出，可以直接用 cout « a 指令，在用 cout 输出数组时，从第一个字符开始，一直输出，直到遇到 “
0 “，如果最初数组赋值没有定义 “
0”，将字符输出之后，会继续输出乱码。所以为了不输出乱码，数组一定要以 “
0” 结尾。

字符数组的输出与普通数字数组不一样，若数字数组这样输出是错误的。

1.4 字符串输入输出

1.4.1 用 cin 输入字符串

在用 cin 输入字符串时，会把空格和回车当作不同字符串的间隔标志。

1.4.2 用 cin.get() 函数输入字符串

此时的 cin.get(ch,10,‘
n’) 函数需要加三个参数。ch: 字符数组的名字；10: 从缓存区读 10-1=9 个字符；‘

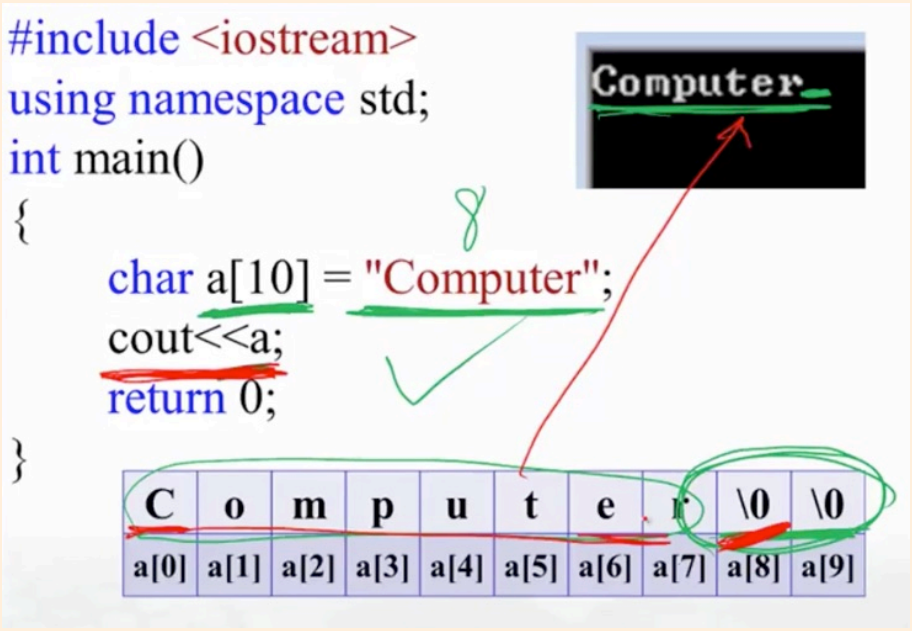


图 5: 用 cout 输出字符示例

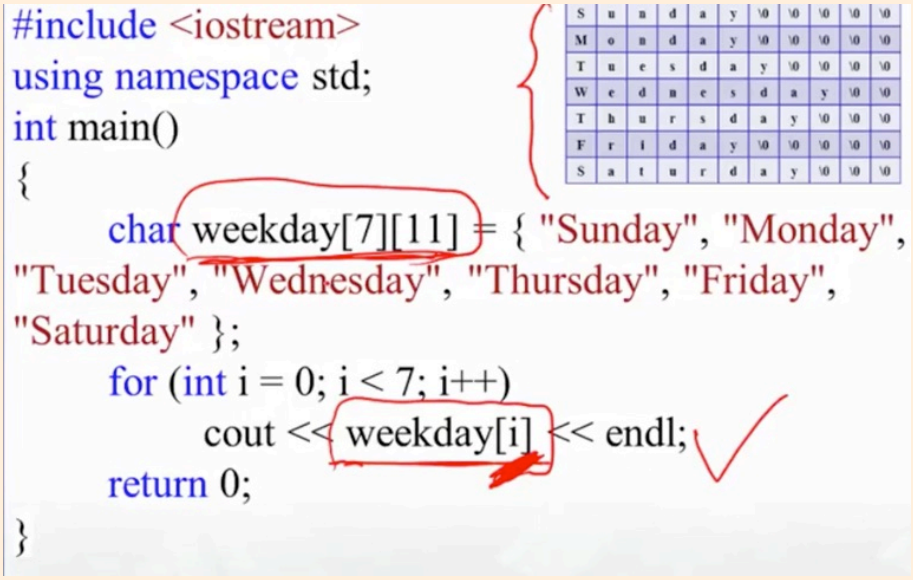


图 6: 二维字符数组输出方法

n' 表示换行。读取 9 个字符，赋给指定的字符数组。如果遇到了指定的终止字符 '
n'，则提前结束读取，否则，一直读取完 9 个字符后结束。

1.4.3 用 cin.getline() 函数输入字符串

与 cin.get() 函数类似。二者的区别，要特别小心处理。getline 遇到终止标志字符时结束，缓冲区指针移到终止标志字符之后。get 遇到终止字符停止读取，指针不移动。

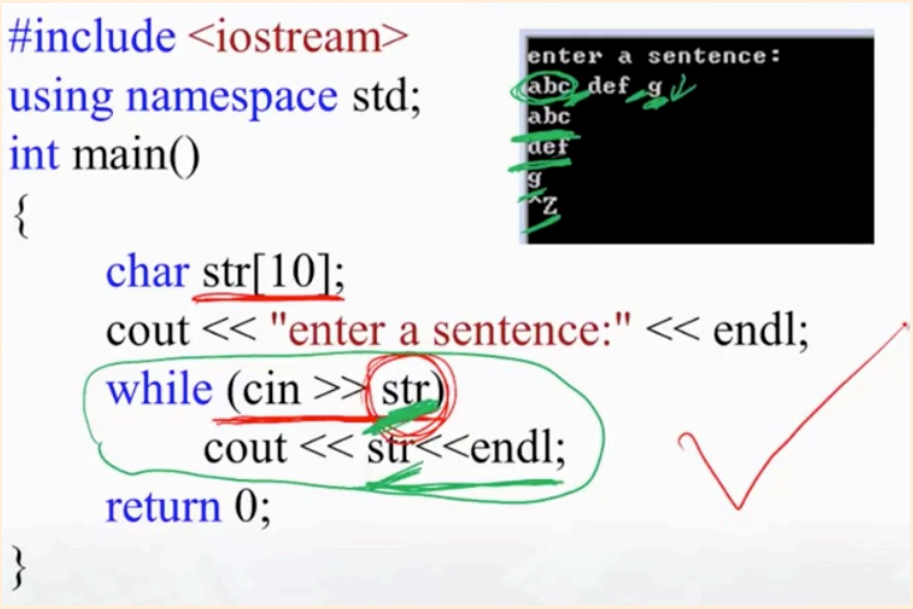


图 7: cin 输入字符串

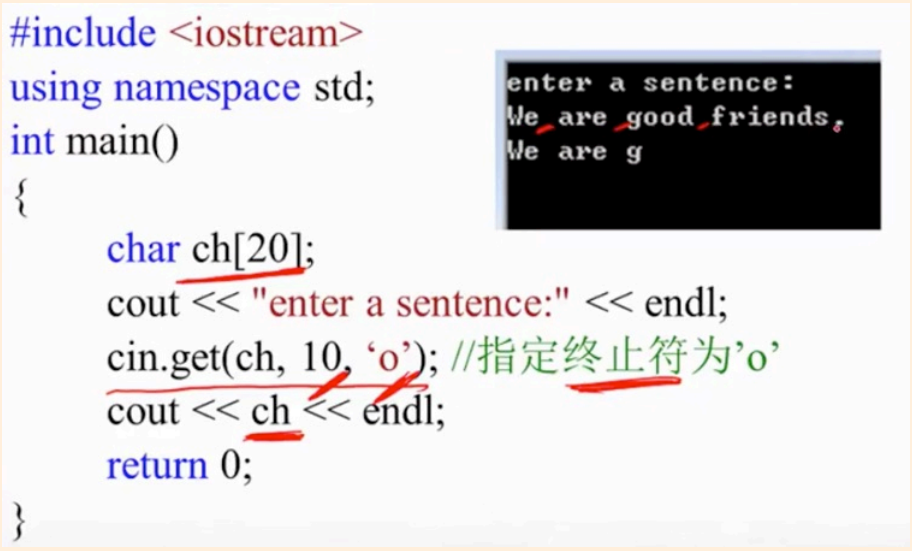


图 8: cin.get() 函数输入

1.5 字符串例题

字符串有很多常用的函数，比如 strcat（拼接两个字符串），strcpy（字符串复制）,strcmp（比较字符串的大小）等，都包含在 include <string> 中。

第 2 章

C 程序中的函数

```
#include <iostream>
using namespace std;
int main()
{
        char ch[20];
        cout << "enter a sentence:" << endl;
        cin.getline(ch, 10, 'o');//指定终止符为'o'
        cout << ch << endl;
        return 0;
}
```

```
enter a sentence:
We are good friends
We are g
```

图 9: cin.get() 函数输入

```
#include <iostream>
using namespace std;
int main()
{
        int len1, len2; char str1[40], str2[40];
        cin.getline(str1, 20); cin.getline(str2, 20);
        for (len1 = 0; str1[len1] != '\0'; len1++);
        for (len2 = 0; str2[len2] != '\0'; len2++);
        if (len1 >= len2)
        {
                for (len2 = 0; str2[len2] != '\0'; len2++)
                        str1[len1++] = str2[len2];
                str1[len1] = '\0';
        }
        else
        {
                for (len1 = 0; str2[len1] != '\0'; len1++)
                        str2[len2++] = str1[len1];
                str2[len2] = '\0';
        }
        cout << str1 << endl;      cout << str2 << endl;
        return 0;
```

图 10: 字符串连接

2.1 函数的定义

2.1.1 常用的函数

1、求平方根：r = sqrt(100.0)；2、求 x 的 y 次方：k = pow(x,y); 3、求一个字符串的长度：i = strlen(str1); 4、比较两个字符串的大小：v = strcmp(str1,str2); 5、把字符串转换为相应的整数：n = atoi(str1);

图 11: 统计单词数

2.1.2 函数的参数

形参：用来辅助进行函数定义的参数。实参：实际运行函数的时候，传递给函数的参数。

2.1.3 需要注意的几点

1、函数的调用，不一定要一个变量来接收它的返回值。函数的三种调用方式见下图：

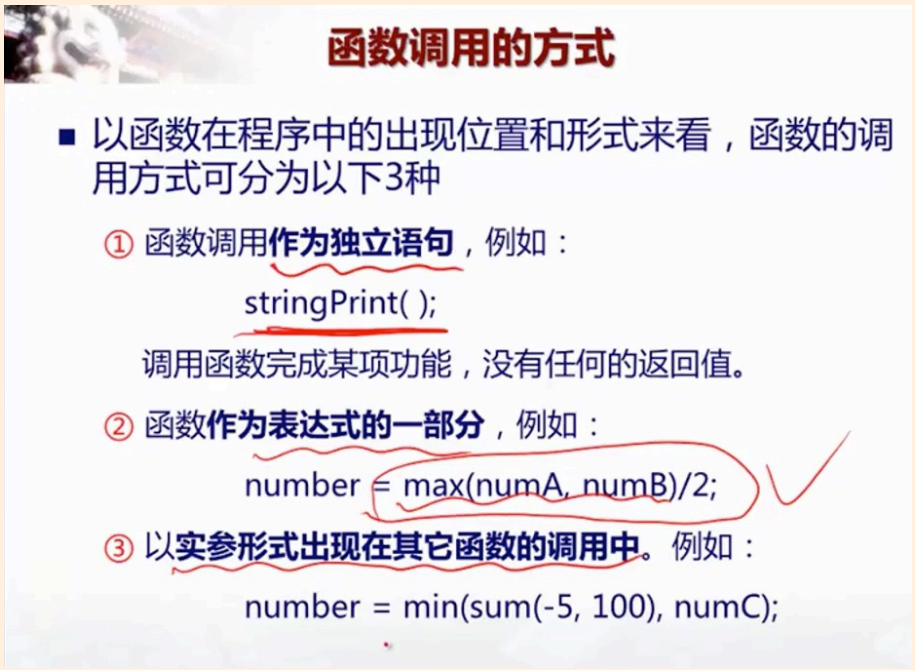


图 12: 三种函数调用方式

2、函数的类型，与其返回值的类型应一致。其次，函数形参与实参的定义方式要明确。形参和实参之间只是值传递，且实参与形参的类型必须相同或兼容。注意：若将函数定义在 main 函数后面，要在 main 函数中声明一下，如：float max(float,float)；这部分是函数的原型（用于对函数的声明），包含函数的返回值类型，函数名和所有参数的类型，仅参数的类型即可。

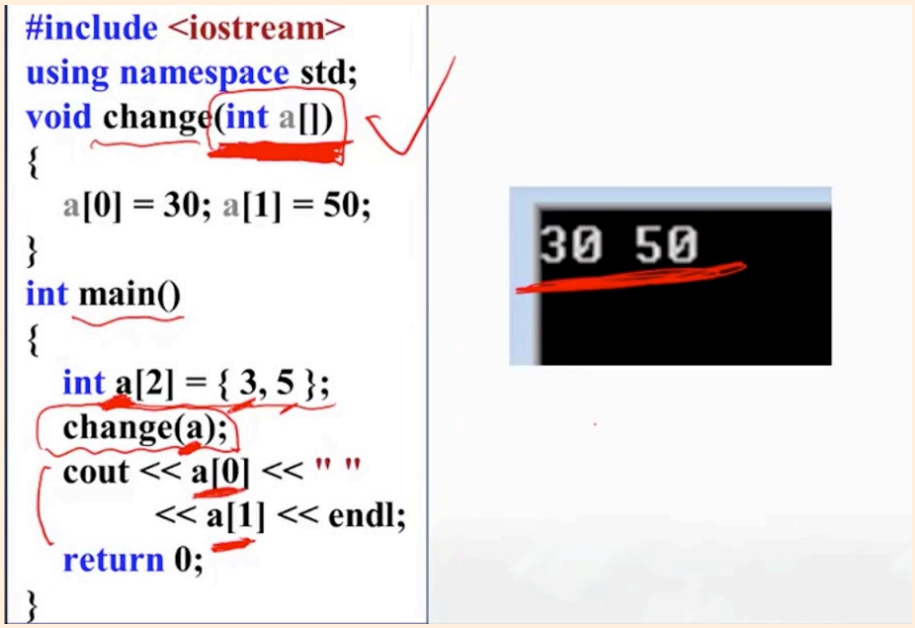


图 13: 简单的函数定义

3、自己定义头文件时，通常使用 include “head.h”，“ ” 与 < > 的区别在于，编译器默认的搜索路径在搜索 < > 内的头文件时，会优先搜索系统函数库所在的目录，“” 会优先搜索当前目录，然后再搜索系统函数库。

2.2 函数的变量

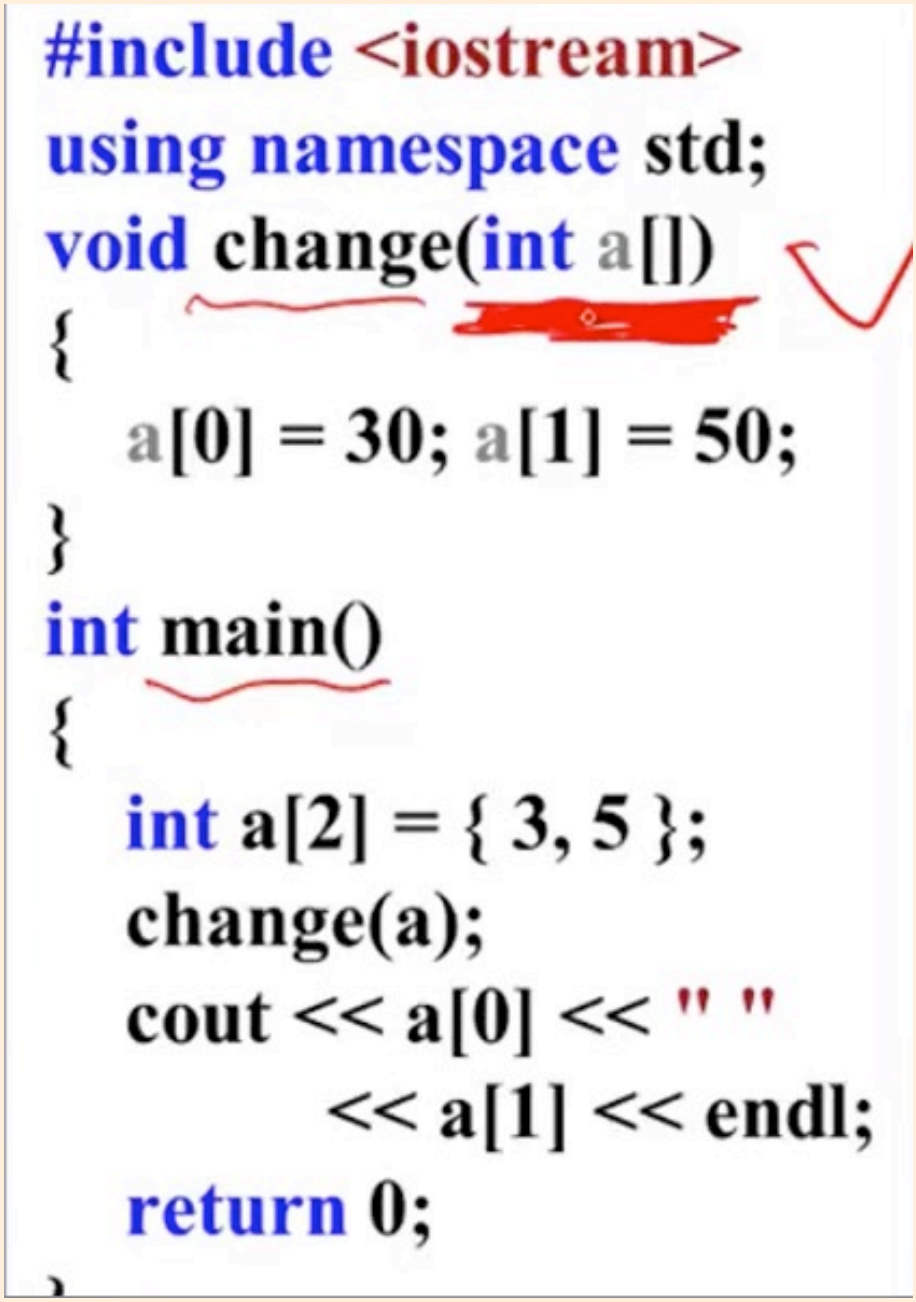
局部变量：在函数内部，或代码块（大括号括起来的代码）内部定义，只在这个函数或块内起作用的变量。

全局变量：在所有函数外定义的变量，它的作用域时从定义变量的位置到程序结束。

局部变量和全局变量的变量名可以一样。当全局变量与局部变量同名时，局部变量在自己的作用域内有效，它将屏蔽同名的全局变量。

2.3 数组与函数

数组做参数：以下图为标准，注意数组做参数的形参实参的形式。注意：我们把数组名字当作实参传递给函数，数组的名字不是变量，而是一个常量，数组的名字代表数组的地址。所以函数会访问数组的地址，修改数组的值。



```
#include <iostream>
using namespace std;
void change(int a[])
{
    a[0] = 30; a[1] = 50;
}
int main()
{
    int a[2] = { 3, 5 };
    change(a);
    cout << a[0] << " "
         << a[1] << endl;
    return 0;
}
```

2.4 函数举例

第 3 章

递归

3.1 函数的递归

3.1.1 递归的定义

函数可以嵌套调用，但是不可以嵌套定义，所有函数，一律平等。

递归：函数直接或间接调用它自身。

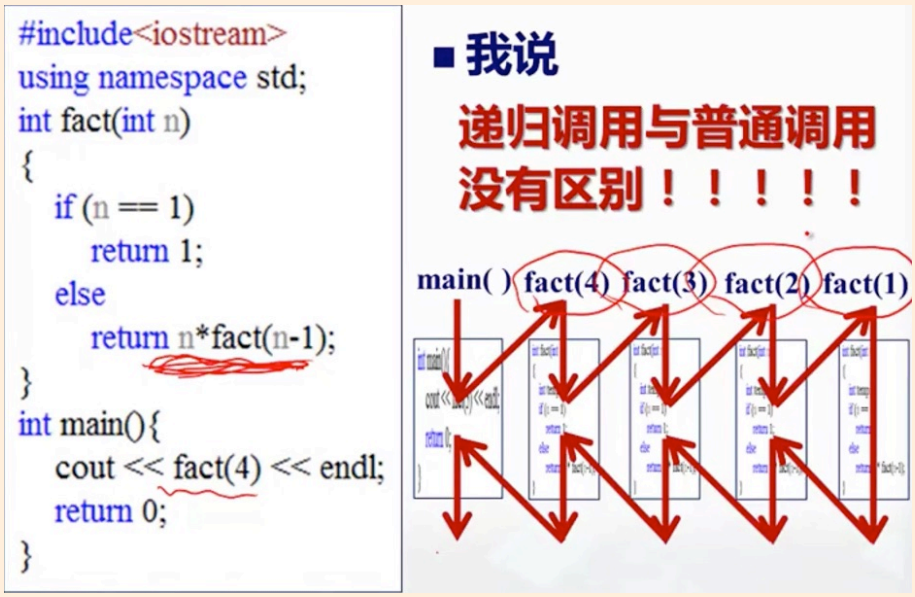


图 14: 递归调用

3.1.2 递归的过程

通过下面的程序，深入理解递归，尤其要注意被调用函数返回时的情况。

3.2 函数举例

3.2.1 递归的应用

1、递归求递推

3.3 函数举例

2、用递归来模拟连续发生的过程。

3.4 函数举例

3、用递归来进行“自动的分析”

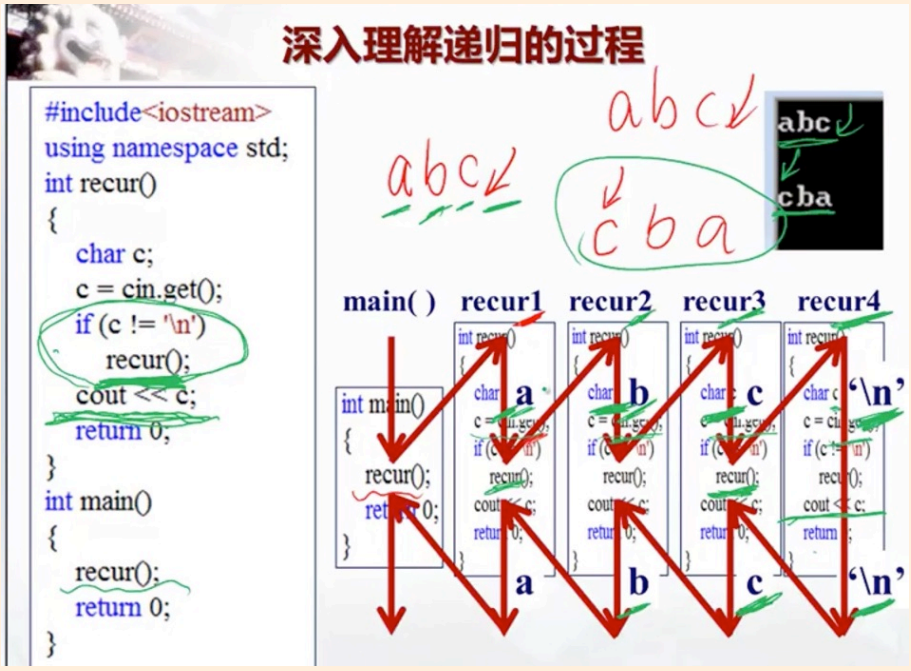


图 15: 递归的过程

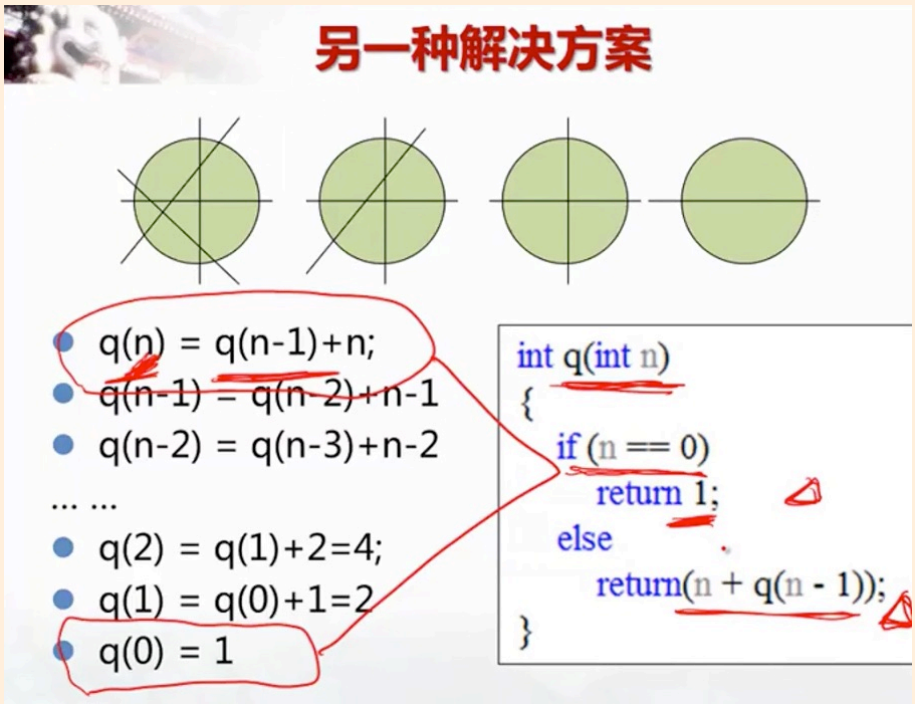


图 16: 递归求递推

3.5 函数举例

3.6 函数举例

图 17: 递归模拟连续过程举例

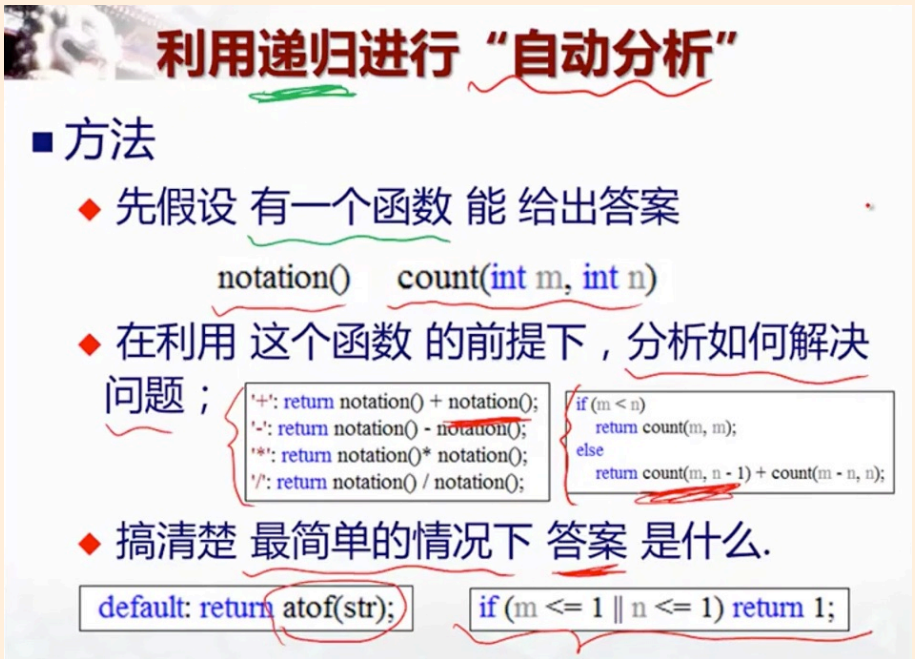


图 18: 用递归来进行“自动的分析”

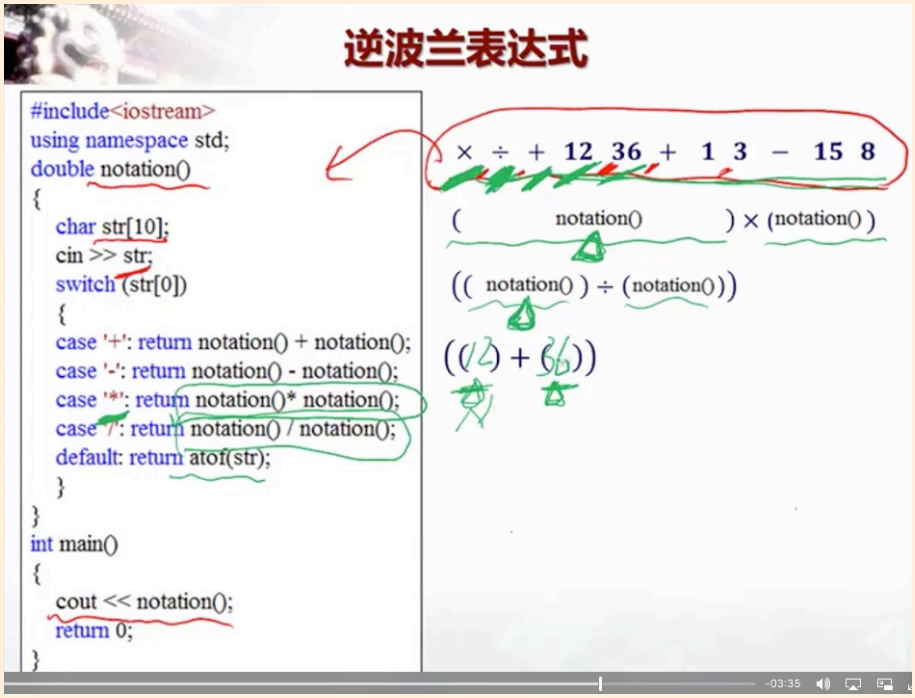


图 19: 经典例题：逆波兰表达式