



《计算概论A》课程 程序设计部分

C++程序中的函数

李 戈

北京大学 信息科学技术学院 软件研究所

2010年11月24日



北京大学

函数 — 一个例子

```
bool checkPrime(int number)
{
    int i, k;
    k = sqrt(number);
    for (i = 2; i <= k; i++)
    {
        if (number % i == 0)           //只要有一个数被出除尽
            return 0;                  //则不是素数。
    }
    return 1;                          //走到这一步，说明没能被除尽
}
```





函数 — 一个例子

```
#include <iostream>
#include <cmath>
using namespace std;
bool checkPrime(int);
int main()
{
    int a;
    cout<<"请输入一个整数"<<endl;
    while(cin>>a)
    {
        if( checkPrime(a) )
            cout<<"是质数"<<endl;
        else
            cout<<"不是质数"<<endl;
    }
    return 0;
}
```



关于函数

■ 函数是C程序的基本构成单位

1. 一个源程序**文件**由一个或多个函数组成。一个源程序文件是一个**编译单位**，而不是以函数为单位进行编译。
2. 一个C程序由一个或多个源程序文件组成。对较大的程序，可以将函数和其他内容**分别放**在若干个源文件中，再由若干源文件组成一个C程序。
3. C程序的执行**从main函数开始**，调用其他函数后流程**回到main函数**，在main函数中结束整个程序的运行。main函数是系统定义的。
4. 所有函数都是**平行的**，即**函数不能嵌套定义**，函数间可以互相调用，但不能调用main函数。





关于函数

- ◆ 函数的定义
- ◆ 函数的类型和返回值
- ◆ 函数的原型和声明
- ◆ 函数的参数和调用
- ◆ 局部变量和全局变量



函数定义

1、获取参数并返回值，例如：

```
int bigger(a, b)
{
    int a, b;
    return (a>b?a:b);
}
...
x = bigger(1, 100)
...
```

形式参数
(形参)

实际参数
(实参)

```
int bigger(int a, int b)
{
    return (a>b?a:b);
}
...
x = bigger(1, 100)
...
```

调用不是void型的函数，必须由return语句返回一个值



函数定义

2、获取参数但是不返回值，例如：

```
void delay(long a)
```

```
{  
    for(int i = 1;i <= a;i++);    //延迟  
}
```

3、没有参数也没有返回值

```
void message()
```

```
{  
    cout << “This is a message.” << endl;  
    return;  
}
```

void 函数也可以用return
语句，但不带参数



北京大学



函数定义

4、没有参数但是有返回值

```
int geti( )  
{   int n;  
    cout<<“input a integer:”<<endl;  
    cin>>n;  
    return n;  
}
```

无论函数完成什么功能，语句有多少，最多只能由return语句返回一个值，叫做函数值。





函数的类型

■ 函数的类型 是指 函数返回值的数据类型

```
int bigger(float, float);  
int main()  
{ float a,b,c;  
  cin>>a>>b;  
  c = bigger(a, b);  
  cout<<“The bigger is ”<<c;  
  return 0;  
}
```

```
int bigger(float x, float y)  
{ return ( x>y?x:y ) ;} //返回时就变成整数了。
```

若函数类型与return语句中表达式的值不一致，则以函数类型为准，系统自动进行类型转换。

运行情况

1.5 2.5

The bigger is 2





函数的原型 及 函数声明

■ 函数原型

- ◆ 由函数的返回类型、函数名以及参数表构成的一个符号串，其中参数可不写名字。

bool checkPrime (int)

■ 函数的声明

- ◆ 函数在使用前都要声明，除非被调用函数的定义部分已经出现在主调函数之前。
- ◆ 在C语言中，函数声明就是函数原型。



bool checkPrime(int); //函数的**声明**

int main()

{ int a;

cout << “请输入一个整数” << endl; cin >> a;

if(checkPrime(a)) //调用函数时给出的参数是**实际参数**

cout << “是质数” << endl;

else

cout << “不是质数” << endl;

return 0;

}

bool checkPrime(int num) //函数定义时给出的参数是**形式参数**

{ int i,k;

k = sqrt(num);

for (i = 2; i <= k; i++)

{ if (num % i == 0) return 0; }

return 1;

}



函数调用的方式

- 以函数在程序中的出现位置和形式来看，函数的调用方式可分为以下3种

① 函数调用作为独立语句，例如：

```
stringPrint( );
```

调用函数完成某项功能，没有任何的返回值。

② 函数作为表达式的一部分，例如：

```
number = max(numA, numB)/2;
```

③ 以实参形式出现在其它函数的调用中。例如：

```
number = min(sum(-5, 100), numC);
```





函数调用的过程

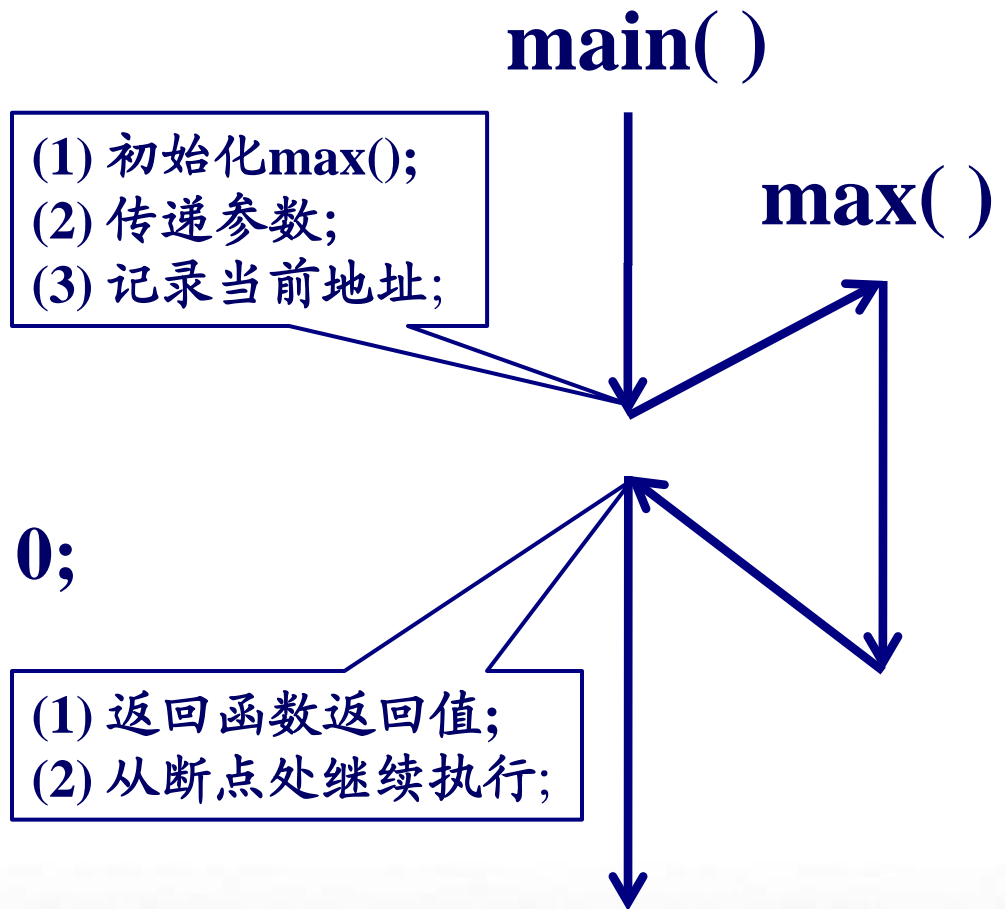
■ 一个函数调用的执行过程可以分为3个阶段:

- ① 首先把实参值传入被调用函数形参的对应单元中，中断主调函数当前的执行，并且保存返回地点(称为断点)。
- ② 执行被调用函数语句，直到**return** 语句返回。若被调用函数中没有**return** 语句，则直到其全部语句执行完毕后自动返回到位于主调函数中的断点处。
- ③ 从保存的断点处，主调函数继续执行其他剩余语句。



函数调用示例

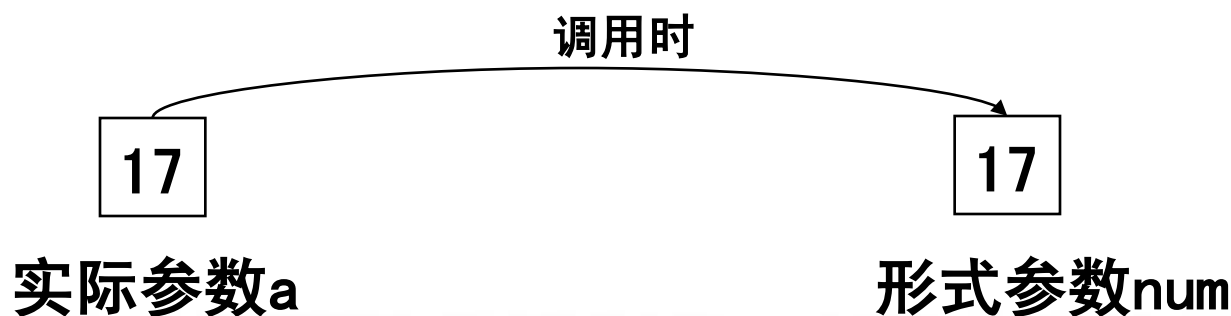
```
int max(int a, int b){  
    int c;  
    c = a > b ? a : b;  
    return(c);  
}  
  
int main( ) {  
    int x = 8, y = 5, z = 0;  
    z = max(x, y);  
    cout << z << endl;  
    return 0;  
}
```



函数的参数与函数调用

■ 参数的传递

- ◆ 实参与形参具有不同的存储单元，实参与形参变量的数据传递是“值传递”；
- ◆ 函数调用时，系统给形参分配存储单元，并将实参对应的值传递给形参；



- ◆ 实参与形参的类型必须相同或可以兼容；





```
void change(int a, int b)
```

```
{
```

```
    a=10; b=20;
```

```
    cout<<a<<b;
```

```
}
```

```
int main ( )
```

```
{
```

```
    int a=2, b=3;
```

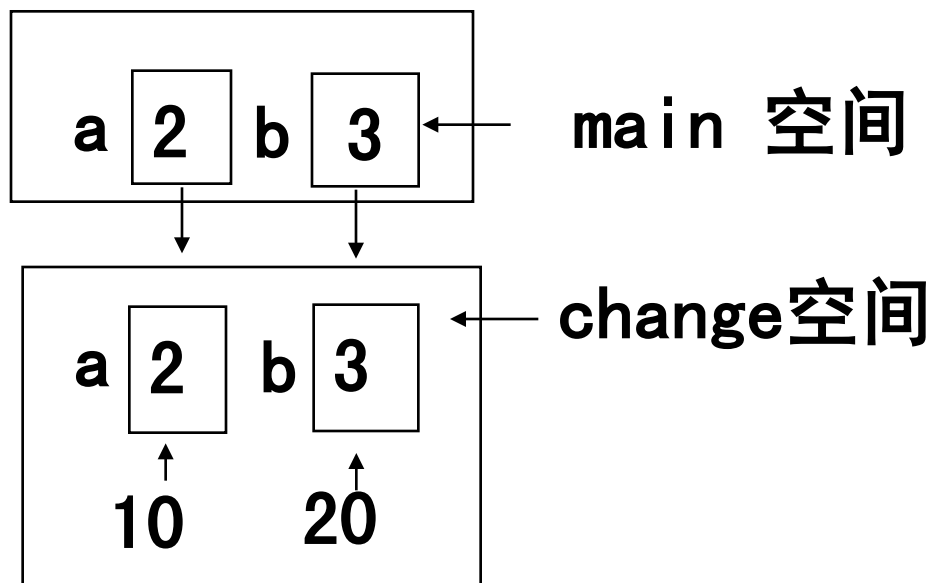
```
    cout<<a<<b;
```

```
    change(a, b);
```

```
    cout<<a<<b;
```

```
    return 0;
```

```
}
```



输出结果

2	3
10	20
2	3



函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
void exchange( int x, int y)
{
    int p;
    if (x < y)
    { p = x; x = y; y = p; }
}
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

程序运行结果？



北京大学

函数的参数与函数调用

```
#include<iostream>
using namespace std;
void exchange( int x, int y)
{
    int p;
    if (x < y)
    { p = x; x = y; y = p; }
}
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

main()

a b
[3] [5]

cout<<a
 <<b
 <<endl;

exchange()

x y
[3] [5]



x y
[5] [3]



北京大學



函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
void exchange( int a, int b)
{
    int p;
    if (a<b)
    { p = a; a = b; b = p; }
}
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

程序运行结果？



北京大学

函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
void exchange( int a, int b)
{
    int p;
    if (a<b)
    { p = a; a = b; b = p; }
}
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

main()

a b
[3] [5]

cout<<a
 <<b
 <<endl;

exchange()

a b
[3] [5]



a b
[5] [3]



北京大學



局部变量与全局变量

■ 根据变量在程序中作用范围的不同，可以将变量分为：

◆ 局部变量：

在函数内或块内定义，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，它的作用域是从定义变量的位置开始到本程序文件结束。





局部变量

- 局部变量：在函数内部定义的变量叫局部变量，它在本函数内有效。

```
float f1(int a){  
    int b, c;  
    ...  
}    //a, b, c有效  
char f2(int x, int y){  
    int i, j;  
    ...  
}    //x, y, i, j有效
```

```
int main( ){  
    int y, m, n;  
    ...  
    f1(y);  
    f2(m, n);  
}    // y, m, n有效
```

说明：

形参是局部变量



函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
void exchange( int x, int y)
{
    int p;
    if (x < y)
    { p = x; x = y; y = p; }
}
```

x, y 的势力范围

```
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

a, b 的势力范围





函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
void exchange( int a, int b)
{
    int p;
    if (a < b)
    { p = a; a = b; b = p; }
}
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```





函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
```

```
void exchange( int a, int b)
{
    int p;
    if (a < b)
    { p = a; a = b; b = p; }
}
```

a, b 的势力范围

```
int main()
{
    int a = 3, b = 5;
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

a, b 的势力范围



全局变量

```
int p=1, q=5; /*外部变量*/
float f1(int a){
    int b, c;
    p++;
}
char c1, c2; /*外部变量*/
char f2(int x, int y); {
    int i, j;
    cin >> c1 >> c2;
}
int main(){
    int m, n;
    cin >> n;
    f1(6);
    m = 2*p + q - n;
    cout << m << c1 << c2;
    return 0; }
```

全局变量
c1, c2的
作用范围

全局变量
p, q的作
用范围



函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
int a = 3, b = 5;
void exchange( )
{
    int p;
    if (a < b)
        { p = a; a = b; b = p; }
}
int main()
{
    exchange( );
    cout<<a<<" "<<b<<endl;
    return 0;
}
```





函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
```

```
int a = 3, b = 5;
```

```
void exchange( )
```

```
{
```

```
    int p;
```

```
    if (a < b)
```

```
    { p = a; a = b; b = p; }
```

```
}
```

```
int main()
```

```
{
```

```
    exchange(a, b);
```

```
    cout<<a<<" "<<b<<endl;
```

```
    return 0;
```

```
}
```

a, b 的势力范围





函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
int a = 3, b = 5;
void exchange( int a, int b)
{
    int p;
    if (a < b)
        { p = a; a = b; b = p; }
}
int main()
{
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```



函数的参数与函数调用

```
#include<iostream>
using namespace std;
int a = 3, b = 5;
void exchange( int a, int b)
{
    int p;
    if (a < b)
    { p = a; a = b; b = p; }
}
int main()
{
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

a, b 的势力范围

a, b 的势力范围

a, b 的势力范围



清华大学

函数的参数 与 函数调用

```
#include<iostream>
using namespace std;
int a = 3, b = 5;
void exchange(int a, int b)
{
    int p;
    if (a < b)
    { p = a; a = b; b = p; }
}
int main()
{
    exchange(a, b);
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

■ 当全局变量与局部变量同名时，局部变量将在自己作用域内有效，它将屏蔽同名的全局变量。





全局变量 的说明

■ 不在必要时不要使用全局变量

- ◆ 在程序的全部执行过程中都 **占用存储单元**。
- ◆ 过多使用全局变量，程序的可读性变差
- ◆ 增加了函数之间的“关联性”，降低了函数的独立性，使函数可移植性降低

■ 特别注意：

- ◆ 如果在同一个源文件中，全局变量与局部变量同名，则在 **局部变量的作用范围内**，全局变量 **不起作用**。





函数 — 一个例子

```
#include <iostream>
#include <cmath>
using namespace std;
bool checkPrime(int);
int main()
{
    int a;
    cout<<"请输入一个整数"<<endl;
    while(cin>>a)
    {
        if( checkPrime(a) )
            cout<<"是质数"<<endl;
        else
            cout<<"不是质数"<<endl;
    }
    return 0;
}
```

```
bool checkPrime(int number)
{
    int i, k;
    k = sqrt(number);
    for (i = 2; i <= k; i++)
    {
        if (number % i == 0)
            return 0;
    }
    return 1;
}
```





函数 — 一个例子

```
#include <iostream>
#include <cmath>
using namespace std;
bool checkPrime(int);
int a = 0;
int main()
{
    cout<<"请输入一个整数"<<endl;
    while(cin>>a)
    {
        if( checkPrime( ) )
            cout<<"是质数"<<endl;
        else
            cout<<"不是质数"<<endl;
    }
    return 0;
}
```

```
bool checkPrime( )
{
    int i, k;
    k = sqrt( a );
    for (i = 2; i <= k; i++)
    {
        if (a % i == 0)
            return 0;
    }
    return 1;
}
```





高内聚 低耦合

■ 结构化程序设计的重要思想

◆ 高内聚

- 函数的全部功能在函数内部完成
- 所有操作对象存放在函数的存储空间内

◆ 低耦合

- 减少函数之间的关联关系
- 函数之间的交互清晰可见





日历问题

■ 问题描述

- ◆ 在我们现在使用的日历中, 闰年被定义为能被4整除的年份, 但是能被100整除而不能被400整除的年是例外, 它们不是闰年。
 - 例如: 1700, 1800, 1900和2100不是闰年, 而1600, 2000 和2400是闰年。
- ◆ 给定从公元2000年1月1日开始逝去的天数, 请编写程序给出这一天是哪年哪月哪日星期几。





日历问题

■ 输入输出要求

- ◆ 输入一个正整数，表示从2000年1月1日开始已经过去的天数。
- ◆ 对输入的每个天数，输出一行，该行包含对应的日期和星期几。格式为：

"YYYY-MM-DD DayOfWeek"

其中 "DayOfWeek" 必须是：Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday.

- ◆ 输入最后一行是-1，不必处理。可以假设结果的年份不会超过9999。



北京大学



日历问题

(1) 计算星期几

```
int get_dayofweek()  
{  
    int dayofweek;  
    dayofweek = days % 7;  
    return dayofweek;  
}  
char week[7][10] = {"Saturday", "Sunday",  
    "Monday", "Tuesday", "Wednesday",  
    "Thursday", "Friday"};
```





日历问题

(2) 计算年数

```
int get_year( )
{
    int i = 2000, leap_year;
    while(1){
        leap_year = (i % 4 == 0 && i % 100 != 0 || i % 400
== 0);
        if(leap_year==1&&days>=366)
        { days = days - 366;    i++; continue;}
        else if(leap_year==0&&days>=365)
        {days = days - 365;    i++; continue;}
        else
            break;
    }
    return i;
}
```



日历问题

(3) 计算月份

```
int get_month(int leap_year)
{
    int pmonth[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int rmonth[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    int j = 0;
    while(1){
        if(leap_year==1 && days>=rmonth[j])
        {
            days=days-rmonth[j];
            j++;
        }
        else if(leap_year==0 && days>=pmonth[j])
        {
            days = days-pmonth[j];
            j++;
        }
        else break;
    }
    return ++j;
}
```



```
#include<iostream>
using namespace std;
int days;
int get_dayofweek();
int get_year();
int get_month(int);
int main()
{
    int year, month, dayofweek; int leap_year;
    char week[7][10] = {"Saturday", "Sunday", "Monday", "Tuesday",
        "Wednesday", "Thursday", "Friday"};
    while ((cin>>days)&& days != -1) {
        dayofweek=get_dayofweek();
        year = get_year();
        leap_year = (year%4==0&&year%100!=0||year%400==0);
        month = get_month(leap_year);
        cout<<year<<"-"<<month<<"-"<<++days<<" "<<week[dayofweek];
    }
    return 0;
}
```



好好想想,有没有问题?

谢谢!



清华大学