

例：编程求出小于n的所有素数（质数）

```
#include <iostream>
using namespace std;
int main()
{
    bool Compute_Prime(int n, ...);
    int n;
    cout << "请输入一个正整数: "
    cin >> n; //从键盘输入一个正整数
    for (int i=2; i<n; i++) //循环：分别判断2、3、...、n-1是否为素数
    {
        int j=2;
        while (j < i && i%j != 0) //循环：分别判断i是否能被2 ~ i-1整除
            j++;
        if (j == i) //i是素数
            cout << i << " ";
    }
    cout << endl;
    return 0;
}
```

4.2 C/C++函数

郭 延 文

2019级计算机科学与技术系

C/C++函数

- ▶ 函数是C/C++提供的用于实现子程序的语言成分。

- ▶ 函数的定义：

<返回值类型> **<函数名>** (**<形式参数表>**)

{

<函数体>

}

- ▶ **<函数名>**：用于标识函数的名字，用标识符表示
- ▶ (**<形式参数表>**)：描述函数的形式参数，由0,1或多个形参说明（用逗号隔开）构成，形参说明的格式为：

 <类型> <形参名>

- ▶ **<返回值类型>**：描述了函数返回值的类型

- ▶ 可以为任意的C++数据类型

- ▶ 当返回值类型为void时，它表示函数没有返回值

函数体

- ▶ **<函数体>**：为一个**<复合语句>**，用于实现相应函数的功能
 - ▶ 函数体内可以包含**return**语句，格式为：
 - **return <表达式>;**
 - **return;**
 - ▶ 当函数体执行到**return**语句时，函数立即返回到调用者。如果有返回值，则把返回值带回给调用者
 - ▶ 如果**return**中的**<表达式>**的类型与函数**<返回值类型>**不一致，则进行隐式类型转换，基本原则为：把**<表达式>**转成**<返回值类型>**



<返回值类型> 造型函数 (面包胚子>)

```
{  
    return: 造型的生面包;  
}
```



造型

输入: 面包胚子

输出: 造型的生面包

<返回值类型> 烘焙函数 (造型的生面包>)

```
{  
    return: 香喷喷的面包;  
}
```



烘焙

输入: 造型的生面包

输出: 香喷喷的面包

函数的例子：求n的阶乘

```
int factorial(int n)  // 返回值 函数名
                      // 形参n: 处理的数据对象
                      // 要有良好的命名意识!
                      // 例如: compute_path (); is_prime();
{
    // 函数体
    int i, k=1;
    for (i=2; i<=n; i++)
        k*= i;
    return k;
}
```

下一个例子：求x的n次幂



double power(double x, int n) // 求x的n次幂

{

if (x == 0) return 0;

// 比较两个浮点数据是否相等:

// 更鲁棒的写法: if(fabs(a-b)<1e-6)

double product=1.0;

if (n >= 0)

while (n > 0)

{ product *= x;

n--;

}

else

while (n < 0)

{ product /= x;

n++;

}

return product;

}



main函数

- ▶ 每个C++程序都要定义一个名字为main的函数，C++程序的执行是从main开始的。对于函数main，其返回值类型为int，例如：

```
int main()
{
    .....
    ... return -1;
    .....
    return 0;
}
```

- ▶ 一般情况下，返回0表示程序正常结束；返回负数（如-1）表示程序非正常结束。

函数返回值（通过return返回）的作用

- ▶ 确定函数是否正确执行而设置返回值

- ▶ main函数 -1,0

- ▶ 带回计算结果

- ▶ int max(int a,int b);

- ▶ 确定被调用函数的运行状态;
根据返回值执行下面的步骤

- ▶ 返回给调用函数的结果,用于函数功能的实现和通信

```
int func(...)  
{  
    if(case1)  
        return 1;  
    else if (case2)  
        return 2;  
    else if (case3)  
        return 3;  
    else  
        return 0;  
}  
// 根据函数调用的返回值,  
// 知道f中发生了什么:  
// 是case1, 2, 3, or nothing!
```

函数返回值（通过return返回）的作用

例如：一只体温计函数

```
int Fthermometer(...)
```

```
// 测量体温的函数
```

医生（调用者）根据

Fthermometer()返回的温度，

确定下一步的应对措施



函数返回值（通过return返回）的作用

```
int f(...)  
{  
    return ...;  
}
```

```
int main()  
{  
    int k = f();  
    if (k == 0)  
        ... ..  
    else if (k == ..)  
        ... ..  
}
```

返回值作用:

1. 带回计算结果

2. 根据返回值执行下面的步骤



函数调用

做甜点

和面

输入：面粉，水

输出：面团



造型

输入：面包胚子

输出：造型生面包



配料

输入：面团

输出：面包胚子



烘焙

输入：造型的生面包

输出：香喷喷的面包



► 每个“功能模块”写为一个函数，供主函数或其他函数调用

函数的调用

- ▶ 对于定义的一个函数，必须要调用它，它的函数体才会执行。
 - ▶ 除了函数main外，程序中对其它函数的调用都是从main开始的。main一般是由操作系统来调用。
 - ▶ 函数调用的格式如下：
 - <函数名>(<实在参数表>);
 - ▶ <实在参数表>由零个、一个或多个表达式构成（逗号分割）
 - ▶ 实参的个数和类型应与相应函数的形参相同
-

函数调用的例子

// 函数定义

int factorial(int n) // 求n的阶乘, ()中n为形参

{

int i,f=1;

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

f *= i;

return f;

}



函数调用的例子

.....

```
int main()
{ int x;
  cout << "请输入一个正整数: ";
  cin >> x;
  cout << "Factorial of " << x << " is "
        << factorial(x) //调用阶乘函数, x是实参
        << endl;
  return 0;
}
```



.....

```
int main()
{ double a;
  int b;
  cout << "请输入a和b: ";
  cin >> a >> b;
  cout << a << "的" << b << "次方是: "
       << power(a,b) << endl;
  return 0;
}
```



函数调用的执行过程

- ▶ 计算实参的值（对于多个实参，C++没有规定计算次序）；
- ▶ 把实参分别传递给被调用函数的形参；
- ▶ 执行函数体；
- ▶ 函数体中执行return语句返回函数调用点，调用点获得返回值（如果有返回值）并执行调用之后的操作。
- ▶ 可以把有返回值的函数调用作为操作数放在表达式中参加运算，例如：

`x+power(x,y)*z;`

看例题

.....

```
int main()
{   double a;
    int b;
    cout << "请输入a和b: ";
    cin >> a >> b;
    cout << a << "的" << b << "次方是: "
         << power(a,b) << endl;
    return 0;
}
```



double power(double x, int n) // 求x的n次幂

```
{  
    if (x == 0)                                // 比较两个浮点数据是否相等:  
        return 0;                             // 更鲁棒的写法: if(fabs(a-b)<1e-6)
```

```
    double product=1.0;  
    if (n >= 0)  
        while (n > 0)  
        {  
            product *= x;  
            n--;  
        }  
    else  
        while (n < 0)  
        {  
            product /= x;  
            n++;  
        }  
    return product;
```

power()函数的定义
写在main()函数后面行吗?

```
}
```



形参 V.S. 实参

- ▶ **形参**：定义函数的时候，为了定义函数功能，代表函数处理的“一般意义上的”数据对象，是**处理对象**的“抽象”
 - ▶ **实参**：实际调用函数的时候，确定的函数**处理对象**的实际数据
-

函数声明

- ▶ 程序中调用的所有函数都要有定义；如果函数定义在本源文件中使用点之后或在其它文件（如：C++的标准库）中，则在调用前需要对被调用的函数进行声明。

- ▶ 函数声明的格式如下：

`extern <返回值类型> <函数名>(<形式参数表>); //函数原型`

- ▶ 在函数声明中，<形式参数表>中可以只列出形参的类型而不写形参名

- ▶ `extern` 可以省略。

```
//file1.cpp
```

```
int x=0; //定义
```

```
extern void f(); //声明
```

```
extern int g(int); //声明
```

```
extern int y; //声明
```

```
int main() //定义
```

```
{
```

```
    y = x + 2;
```

```
    f(); //调用
```

```
    y = g(x); //调用
```

```
    return 0;
```

```
}
```

```
int y=0; //定义
```

```
void f() //定义
```

```
{
```

```
    x = y + 1;
```

```
}
```

```
//file2.cpp
```

```
extern int x,y; //声明
```

```
int g(int i) //定义
```

```
{
```

```
    int z; //定义
```

```
    z = x + y;
```

```
    return z+i;
```

```
}
```

例：用函数实现求小于n的所有素数

（要求：每行打印6个）。

```
#include <iostream>
#include <cmath>
using namespace std;
bool is_prime(int n); //函数声明
void print_prime(int n, int count); //函数声明
int main()
{   int i,n,count=1;
    cout << "请输入一个正整数: "
    cin >> n; //从键盘输入一个正整数
    if (n < 2) return -1;
    cout << 2 << ","; //输出第一个素数
    for (i=3; i<n; i+=2)
    {   if (is_prime(i))
        {   count++;
            print_prime(i,count);
        }
    }
    cout << endl;
    return 0;
```



```
bool is_prime(int n) //函数定义
{
    int i, j, k=sqrt(double(n));
    for (i=2, j=k; i<=j; i++)
        if (n%i == 0)
            return false;
    return true;
}
```

```
void print_prime(int n, int count) //函数定义
{
    cout << n << ',';
    if (count % 6 == 0)
        cout << endl;
}
```



参数传递

做甜点

和面

输入：面粉，水

输出：面团



造型

输入：面包胚子

输出：造型生面包



配料

输入：面团

输出：面包胚子



烘焙

输入：造型的生面包

输出：香喷喷的面包



► 每个函数（功能模块）都有处理的对象，实际调用时才确定其值

例：用函数实现求小于n的所有素数（要求： 每行打印6个）。

```
#include <iostream>
#include <cmath>
using namespace std;
bool is_prime(int n);//函数声明
void print_prime(int n, int count);//函数声明
int main()
{   int i,n,count=1;
    cout << "请输入一个正整数: "
    cin >> n; //从键盘输入一个正整数
    if (n < 2) return -1;
    cout << 2 << ","; //输出第一个素数
    for (i=3; i<n; i+=2)
    {   if (is_prime(i)) // 值传递
        {       count++;
                print_prime(i, count); // 值传递
        }
    }
    cout << endl;
    return 0;
```



函数调用的执行过程

- ▶ 计算实参的值；
- ▶ 把实参分别传递给被调用函数的形参；
- ▶ 执行函数体；
- ▶ 函数体中执行return语句返回函数调用点，调用点获得返回值（如果有返回值）并执行调用之后的操作。



函数的参数传递

- ▶ C++提供了两种参数传递机制：
 - ▶ 值传递
 - ▶ 把实参的值赋值给形参。
 - ▶ 地址或引用传递
 - ▶ 把实参的地址赋值给形参。
- ▶ C++默认的参数传递方式是值传递。



值传递

- ▶ 在函数调用时，采用类似变量初始化的形式把实参的值传给形参。
- ▶ 函数执行过程中，通过形参获得实参的值，
- ▶ 函数体中对形参值的改变不会影响相应实参的值。



值参数传递的例子

//函数main调用函数power计算 a^b

```
#include <iostream>
```

```
using namespace std;
```

```
double power(double x, int n);
```

```
int main()
```

```
{ double a=3.0,c;
```

```
  int b=4;
```

```
  c = power(a, b);
```

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

```
  return 0;
```

```
}
```



```
double power(double x, int n)
{ if (x == 0) return 0;
  double product=1.0;
  if (n >= 0)
    while (n > 0)
    { product *= x;
      n--;
    }
  else
    while (n < 0)
    { product /= x;
      n++;
    }
  return product;
}
```



- ▶ 执行main时，产生三个变量（分配内存空间）a、b和c：

a: 3.0 b: 4 c: ?

- ▶ 调用power函数时，又产生三个个变量x、n和product，然后分别用a、b以及1.0对它们初始化：

a: 3.0 b: 4 c: ?

x: 3.0 n: 4 product: 1.0

- ▶ 函数power中的循环结束后（函数返回前）：

a: 3.0 b: 4 c: ?

x: 3.0 n: 0 product: 81.0

- ▶ 函数power返回后：

a: 3.0 b: 4 c: 81.0

内容回顾

- ▶ 函数－实现过程抽象与封装
 - ▶ 基于过程抽象的程序设计－子程序
- ▶ C/C++ 函数
 - ▶ 函数定义
 - ▶ 函数调用
 - ▶ 函数声明
 - ▶ 参数传递



Q & A

