> 函数让代码变得更加简洁。——闫学灿

1. 函数基础

一个典型的函数定义包括以下部分：返回类型、函数名字、由0个或多个形参组成的列表以及函数体。

1.1 编写函数

我们来编写一个求阶乘的程序。程序如下所示：

```
int fact(int val)
{
    int ret = 1;
    while (val > 1)
        ret *= val -- ;
    return ret;
}
```

函数名字是 `fact` ，它作用于一个整型参数，返回一个整型值。 `return` 语句负责结束 `fact` 并返回 `ret` 的值。

1.2 调用函数

```
int main()
{
    int j = fact(5);
    cout << "5! is " << j << endl;

    return 0;
}
```

函数的调用完成两项工作：一是用实参初始化函数对应的形参，二是将控制权转移给被调用函数。此时，主调函数的执行被暂时中断，被调函数开始执行。

1.3 形参和实参

实参是形参的初始值。第一个实参初始化第一个形参，第二个实参初始化第二个形参，依次类推。形参和实参的类型和个数必须匹配。

```
fact("hello");      // 错误：实参类型不正确
fact();             // 错误：实参数量不足
fact(42, 10, 0);    // 错误：实参数量过多
fact(3.14);         // 正确：该实参能转换成int类型，等价于fact(3);
```

形参也可以设置默认值，但所有默认值必须是最后几个。当传入的实参个数少于形参个数时，最后没有被传入值的形参会使用默认值。

1.4 函数的形参列表

函数的形参列表可以为空，但是不能省略。

```
void f1() {/* …. */}            // 隐式地定义空形参列表
void f2(void) {/* … */}         // 显式地定义空形参列表
```

形参列表中的形参通常用逗号隔开，其中每个形参都是含有一个声明符的声明。即使两个形参的类型一样，也必须把两个类型都写出来：

```
int f3(int v1, v2) {/* … */}        // 错误
int f4(int v1, int v2) {/* … */}    // 正确
```

1.5 函数返回类型

大多数类型都能用作函数的返回类型。一种特殊的返回类型是 `void` ，它表示函数不返回任何值。函数的返回类型不能是数组类型或函数类型，但可以是指向数组或者函数的指针。

1.6 局部变量、全局变量与静态变量

局部变量只可以在函数内部使用，全局变量可以在所有函数内使用。当局部变量与全局变量重名时，会优先使用局部变量。

2. 参数传递

2.1 传值参数

当初始化一个非引用类型的变量时，初始值被拷贝给变量。此时，对变量的改动不会影响初始值。

```
#include <iostream>

using namespace std;

int f(int x)
{
    x = 5;
}

int main()
{
    int x = 10;

    f(x);
    cout << x << endl;

    return 0;
}
```

2.2 传引用参数

当函数的形参为引用类型时，对形参的修改会影响实参的值。使用引用的作用：避免拷贝、让函数返回额外信息。

```cpp
#include <iostream>

using namespace std;

int f(int &x)
{
    x = 5;
}

int main()
{
    int x = 10;

    f(x);
    cout << x << endl;

    return 0;
}
```

2.3 数组形参

在函数中对数组中的值的修改，会影响函数外面的数组。

一维数组形参的写法：

```cpp
// 尽管形式不同，但这三个print函数是等价的
void print(int *a) {/* … */}
void print(int a[]) {/* … */}
void print(int a[10]) {/* … */}
```

```cpp
#include <iostream>

using namespace std;

void print(int a[])
{
    for (int i = 0; i < 10; i ++ )
        cout << a[i] << endl;
}

int main()
{
    int a[10];

    for (int i = 0; i < 10; i ++ )
        a[i] = i;

    print(a);

    return 0;
}
```

多维数组形参的写法：

```
// 多维数组中，除了第一维之外，其余维度的大小必须指定
void print(int (*a)[10]) {/* … */}
void print(int a[][10]) {/* … */}
```

```
#include <iostream>

using namespace std;

void print(int a[][10])
{
    for (int i = 0; i < 10; i ++ )
    {
        for (int j = 0; j < 10; j ++ )
            cout << a[i][j] << ' ';
        cout << endl;
    }
}

int main()
{
    int a[10][10];

    for (int i = 0; i < 10; i ++ )
        for (int j = 0; j < 10; j ++ )
            a[i][j] = j;

    print(a);

    return 0;
}
```

3. 返回类型和return语句

`return`语句终止当前正在执行的函数并将控制权返回到调用该函数的地方。`return`语句有两种形式：

```
return;
return expression;
```

3.1 无返回值函数

没有返回值的`return`语句只能用在返回类型是`void`的函数中。返回`void`的函数不要求非得有`return`语句，因为在这类函数的最后一句后面会隐式地执行`return`。

通常情况下，`void`函数如果想在它的中间位置提前退出，可以使用`return`语句。`return`的这种用法有点类似于我们用`break`语句退出循环。

```
void swap(int &v1, int &v2)
{
    // 如果两个值相等，则不需要交换，直接退出
    if (v1 == v2)
        return;
    // 如果程序执行到了这里，说明还需要继续完成某些功能

    int tmp = v2;
    v2 = v1;
    v1 = tmp;
    // 此处无须显示的return语句
}
```

3.2 有返回值的函数

只要函数的返回类型不是 `void` ，则该函数内的每条 `return` 语句必须返回一个值。 `return` 语句返回值的类型必须与函数的返回类型相同，或者能隐式地转换函数的返回类型。

```
#include <iostream>

using namespace std;

int max(int x, int y)
{
    if (x > y) return x;

    return y;
}

int main()
{
    int x, y;
    cin >> x >> y;

    cout << max(x, y) << endl;

    return 0;
}
```

4. 函数递归

在一个函数内部，也可以调用函数本身。

```cpp
#include <iostream>

using namespace std;

int fact(int n)
{
    if (n <= 1) return 1;
    return n * fact(n - 1);
}

int main()
{
    int n;
    cin >> n;

    cout << fact(n) << endl;

    return 0;
}
```