1、函数的基本思想

函数强调代码的**复用性**，即可以作为一个整体被反复使用。
函数可以被其他部分的代码使用，称为**调用**；也可以**被自身调用**，即**递归**。

把可以重复使用的代码封装成一个函数，用以实现特定功能，这种**模块化**的编程方式更符合人的思考方式（**使用工具未必要知道所有原理**），一定程度上减少了重复造轮子的工作量。

2、函数的定义、声明和调用

关于函数的**定义和声明**：
void myfun(){cout<<"This is my fun.";} // 这是定义
void myfun(); // 这是声明
函数的定义和声明在**形式**上的**区别**是：①谁有函数体（即**大括号**）？ ②谁以**分号**结尾？

【如果一个函数的定义写在它被调用之后，则必须在调用前先声明。】
（读类似很绕的话的时候，宜总结成更具体的形式，如：①声明-调用-定义 ②定义-调用）

调用函数时，不可以**写返回类型和参数类型**！但**必须有小括号**，如 myfun()。
即使在**没有实际参数**的情况下，也**必须有**空的小括号，不能漏掉。

3、函数的返回

一般情况下，函数都有**返回类型**（void 也是一种返回类型）。
【特例：第 10 章介绍的构造函数和析构函数没有返回类型】
一个函数如果**没有**返回**值**，那么它的返回**类型**是 **void**。
返回类型**不是 void** 的函数都必须以 **return** 语句结束。
返回类型是 void 的函数也能以 return 语句结束（即 return;）。
如果一个函数的返回类型不是 void，且在不同情况下返回的值不同，那么一定要保证在每个逻辑分支下都有 return 语句。

如果 **return 语句**返回的变量的类型和**函数**的返回类型不同，会进行自动类型转换。

```
函数的定义：
int myfun() {return 6.67;}

函数的调用：
double x;    x = myfun(); // x 被赋值为 6
```

4、函数的参数

当参数为值传递时，被调函数中对形参的修改不影响主调函数中实参的值。
【虽然这句话每个人都懂，但写起代码来还是有很多同学会忘记】
【想想看自己有没有遇到过这种情况，想一下正确的代码和错误的代码分别长什么样子】
如果**实参**和**形参**的**类型不同**，会进行**自动类型转换**。【还记得自动类型转换的规则吗？】

5、变量的作用范围（作用域）

程序块是以**大括号**作为**边界**的一段代码。
在一个程序块中**定义**的变量（即局部变量）**只能**在这个程序块里使用，在该程序块结束时会消亡/释放（即该程序块为该局部变量的作用域）。
【很多同学的作业中都曾经出现过这类错误，大家务必养成检查变量作用域的习惯。】
一个特例是在 for 循环控制行中定义的变量，虽不在大括号里，但属于大括号里的程序块。
【第 7 章介绍的用 new 定义的动态变量**不是**任何程序块中的变量，不 delete 就不会消亡。】

整个源文件可以看作最大的程序块。【提问：全局变量的作用范围是整个源文件吗？】
在内部程序块中**定义**的变量会屏蔽在外部程序块中定义的同名变量。

```
int main(){
    int i = 27;
    for(int i=0; i<10; i++)
        cout << i; //输出 0-9
    cout << i; //输出 27
    return 0;
}
```

```
int main(){
    int i = 27;
    for(i=0; i<10; i++)
        cout << i; //输出 0-9
    cout << i; //输出 10
    return 0;
}
```

【注意避免混淆】如果内部程序块没有定义与外部程序块同名的变量，而是直接修改了在外部程序块中定义的变量，那么其修改结果自然会延续到外部程序块中。
例如 int i=0; for(i=0; i<20; ++i)cout<<i; 在循环结束后，i 的值就是 20。

【四种存储类型是哪四种？分别如何使用？请仔细看课本，务必搞懂。】

6、数组作为函数的参数

当**形参**写成数组形式时（如 int a[ ]），本质上是个指向 int 变量的指针（第 7 章会讲）。
如果**实参**是数组名，那么会把该数组的起始地址赋值给这个形参指针。被调函数可以通过该指针修改主调函数中的数组元素。

将多维数组作为函数的形式参数时，要指定除了第一维以外所有维的大小。
例如三维数组的情况：int a[ ][5][10]，其本质是一个指向 5x10 二维 int 数组的指针。

7、函数的参数带默认值（缺省值）

当函数的参数带默认值时，要注意**默认值的位置**。
在 ①声明-调用-定义 的情况下，参数的默认值写在函数的声明语句里。
在 ②定义-调用 的情况下，参数的默认值写在函数头里。

8、内联函数

内联函数：在编译时用函数体替换函数调用语句，以消除运行时的调用开销。

【内联的优点是既能重用代码又没有调用开销，缺点是会导致编译后的代码膨胀。】
内联函数的定义应该写在调用之前，并在函数头前加上关键字 inline。
【在什么情况下不适合使用内联函数？】

9、函数重载与函数模板

重载函数：函数名相同，但参数表不同。
注意：如果两个函数的返回类型不同但参数表相同，那么它们**不可以**重名。

函数模板不是函数，它在运行时根据**调用**语句中**实参的数据类型**产生一个对应的函数。
【如果多次调用，不同的实参类型会产生不同的函数，相同的实参类型使用同一个函数。】
函数模板在声明和定义时都要加上 template <class T>。
定义多个函数模板时，每个函数模板都要写属于各自的 template <class T>。

如果一个函数模板中有多个不同的参数化的类型，则每种类型对应一个 class，例如：
template <class T, class S, class U>

10、递归函数

在写递归函数时，尤其要注意（1）递归的终止条件与（2）回溯的过程。
一个递归函数的函数体中包括：①递归调用前的处理 ②递归调用 ③递归调用后的处理
上述第①步需检查递归的终止条件是否成立，第③步即回溯时执行的操作。

用递归函数处理数组时，特别注意数组的起始地址和长度在递归过程中的变化。