函数

- 程序设计中三种基本控制结构,**顺序，选择，控制**可以组成任何程序
- 如果相同功能的程序片段在程序不同地方反复出现,就可以将这些片段作为相对独立的整体,用一个标识符给他们起一个名字
- 凡是**程序中出现该程序片段**的地方就可以**简单的写上标识符**就可以
- **这样的程序片段就是函数**
- **使用函数的好处:**
  - 有利于结构化程序的设计
  - 一个复杂的问题划分为多个子问题来解决,如果子问题很复杂,可以继续划分为几个子问题
  - 这样编写的程序结构清晰,便于扩展阅读调试迁移修改
- 主函数和子函数:
  - 主函数 main()一个程序有且只有一个主函数,程序从主函数开始执行
  - 子函数,是除了主函数之外的函数,程序
  - **可以只有主函数而没有子函数,但是不能没有主函数**
  - **子函数不能单独运行**

函数的声明和使用

1. 认识第一个函数:

```
1.  #include<iostream>
2.  using namespace std;
3.  int max2(int, int);
4.  int main()
5.  {
6.      int a = max2(3, 5);
7.      int b = max2(10, -1);
8.      cout << a << " " << b << endl;
9.      return 0;
10. }
11. int max2(int x, int y)
12. {
13.     return (x > y) ? x : y;
14. }
```

- 1.处是函数的声明
- 2.处是函数的定义
- 3.处是函数的使用

2. 函数的定义

```
1.  数据类型 函数名(形式参数表)
2.  {
3.      //函数体执行语句
4.      ...
5.  }
```

```
1.  int max2(int x, int y)
2.  {
3.      return (x > y) ? x : y;
4.  }
```

- 函数名:
  - 是标识符,
  - 主函数必须为 main,
  - 其余函数的名字都是在遵循标识符的取名规则下选取的,
  - 最好帮助记忆
- 形式参数表:
  - 形参表可以是空的
  - 可以有多个形参,中间用逗号隔开
  - 无论有无形参,后面必须有小括号
  - 形参必须有类型声明,可以是变量名,数组名或者指针名
  - 它标识着形参出现的位置应该放一个什么样类型的数据,例如 int x, double y
  - 被调用的时候也就是使用的时候才将实际参数(实参)赋予给形参
- 函数体{. }
  - 用大括号括起来表示若干个语句形成一函数的函数体
  - 函数内的语句决定了函数的功能
  - 如果函数体为空,叫做空函数
  - 函数不允许嵌套定义,不允许在一个函数内定义另一个函数
  - 函数允许嵌套使用
- 返回类型:
  - 函数的返回类型是函数的返回值类型
  - 如果函数不需要返回值就把返回值设置为 void

3. 函数的声明

- **调用函数之前必须声明函数的原型,**例如下面的**第三行**

```
1.  #include<iostream>
2.  using namespace std;
3.  int max2(int, int);
4.  int main()
5.  {
6.      int a = max2(3, 5);
7.      int b = max2(10, -1);
8.      cout << a << " " << b << endl;
9.      return 0;
10. }
11. int max2(int x, int y)
12. {
13.     return (x > y) ? x : y;
14. }
```

- 函数的声明和函数的定义第一行类似,只是多了一个分号,

- 当然,在函数的声明的时候可以不需要指定形参的变量名,但一定要指定形参的数据类型
- 也可以将函数的定义和声明写到一起,这样可以省略函数的声明

```
#include<iostream>
using namespace std;
int max2(int x, int y)
{
    return (x > y) ? x : y;
}
int main()
{
    int a = max2(3, 5);
    int b = max2(10, -1);
    cout << a << " " << b << endl;
    return 0;
}
```

4. 函数的调用

- 主调函数中的参数称为实参,实参应该具有确定的值
- 实参列表应该与函数原型的形参的个数相同,类型匹配
- 实参可以是常量,表达式也可以是由确定值的表达式
- 函数调用可以作为一条语句,这个时候函数是可以没有返回值的

5. 函数的返回值

- -return 表达式;
- 功能是把程序流程从被调函数返回给主调函数,并把表达式的值带回主调函数,实现函数的返回
- 返回值类型就是函数的类型
- 当函数没有返回值的时候,函数可以没有 return 语句,或者 return;直接写分号

函数的应用

组合数的计算方法

- 组合数是指

  组合数C_m^n表示从m个不同的数中任意取出n个数的所有不同可能的情况的数目。

- $$C_m^n = \frac{m!}{(m-n)! \times n!}$$

- 输入输出:

【输入格式】

- 一行：m和n

【输出格式】

- 一行：计算结果

【输入样例】

```
4 2
```

【输出样例】

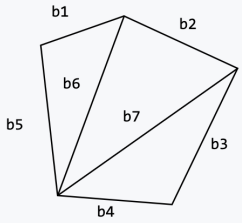
```
6
```

计算五边形的面积

- 问题描述：

【问题描述】

- 给定七条边的长度，计算五边形的面积。



【输入格式】

- 一行：七条边的长度

【输入样例】

```
1 1 1 1 1 1 1
```

【输入格式】

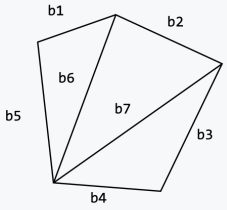
- 一行：计算结果

【输出样例】

```
1.29904
```

- 算法分析

- 五边形的面积是三个三角形面积之和。
- 海伦公式可以根据边长计算三角形的面积。



【海伦公式】

$$p = \frac{a+b+c}{2}$$

$$S = \sqrt{p \times (p-a) \times (p-b) \times (p-c)}$$

判断质数

- 算法描述

【问题描述】

- 写一个布尔函数，判断给定的正整数是否是质数。

【输入格式】

- 一个正整数。

【输入样例】

```
5
```

【输出格式】

- 0表示不是质数，1表示是质数。

【输出样例】

```
1
```

- 算法分析：

- 对于任意正整数i，
- 根据质数定义，从2开始到$\sqrt{i}$，找i的第一个约数，
- 若找到第一个约数，则i必然不是质数。

判断回文数和完全平方数

- 题目：

【问题描述】

- 要把两个数m和n之间所有整数中的回文数和完全平方数都标记出来，如果一个数是回文数，则在数字后面添加标记'@'，如果是完全平方数，则在数字前面添加标记'*'。
- 输入有一行，包含2个用空格隔开的整数m和n。
- 输出m和n之间（包含m和n）的所有整数，回文数后面有'@'标记，完全平方数前面有'*'标记。

【输入样例】

```
110 130
```

【输出样例】

```
110 111@ 112 113 114 115 116 117 118 119 120 *121@ 122 123
124 125 126 127 128 129 130
```

- 问题描述：

判断回文数的代码可以封装成一个函数，通过将一个数各个数位上的数字倒过来，和原来的数进行比较来判断是否回文数。

判断完全平方数的代码可以封装为一个函数，通过判断这个数的平方根的平方是否与这个数相等来判断是否完全平方数。

枚举m和n之间所有的整数，通过判断状态来添加不同的标记。

函数的调用方式

全局变量和局部变量

- **全局变量：**

  全局变量：在所有函数外部声明的变量。
  全局变量的作用域是从变量定义的位置开始到文件结束。
  全局变量的值在程序的整个生命周期内都是有效的。
  全局变量在程序执行的全过程中一直占用内存单元。
  全局变量在定义时若没有赋初值，其默认值为0。

- **局部变量：**

  局部变量：在函数或一个代码块内部声明的变量。
  - ✓ 主函数main中定义的变量是局部变量。
  - ✓ 函数的形参也是局部变量。

  局部变量只能被函数内部或者代码块内部的语句使用。在不同的函数中变量名可以相同，它们分别代表不同的对象，在内存中占据不同的内存单元，互不干扰。
  局部变量的存储空间是临时分配的，当函数执行完毕，局部变量的空间就被释放，其中的值无法保留到下次使用。
  一个局部变量和一个全局变量是可以重名的，在相同的作用域内局部变量有效时全局变量无效。即局部变量可以屏蔽全局变量。
  全局变量初始全部为0；局部变量值是随机的，必须要初始化初值。
  局部变量受栈空间大小限制，无法开辟规模很大的数组。

- 使用全局变量的好处和坏处

【全局变量的好处】

- 使得函数间多了一种传递信息的方式。
- 如果在一个程序中多个函数都要对同一个变量进行处理，就可以将这个变量定义成全局变量。

【全局变量的坏处】

- 增加调试难度。因为多个函数都能改变全局变量的值，不易判断某个时刻全局变量的值。
- 降低程序的通用性。如果将一个函数移植到另一个程序中，需要将全局变量一起移植过去，同时还有可能出现重名问题。

函数的值调用和引用调用

```
#include<iostream>
using namespace std;
void swap(int a, int b)
```

无法交换 a 和 b 的值

```
#include<iostream>
using namespace std;
void swap(int &a, int &b)
```

可以交换 a 和 b 的值