

函数

素数求和

```
scanf("%d %d", &m, &n);
// m=10, n=31;
if ( m==1 ) m=2;
for ( i=m; i<=n; i++ ) {
    int isPrime = 1;
    int k;
    for ( k=2; k<i-1; k++ ) {
        if ( i%k == 0 ) {
            isPrime = 0;
            break;
        }
    }
    if ( isPrime ) {
        sum += i;
        cnt++;
    }
}
printf("%d %d\n", cnt, sum);
```

```
int isPrime(int i)
{
    int ret = 1;
    int k;
    for ( k=2; k<i-1; k++ ) {
        if ( i%k == 0 ) {
            ret = 0;
            break;
        }
    }
    return ret;
}
```

```
scanf("%d %d", &m, &n);
// m=10, n=31;
if ( m==1 ) m=2;
for ( i=m; i<=n; i++ ) {
    if ( isPrime(i) ) {
        sum += i;
        cnt++;
    }
}
printf("%d %d\n", cnt, sum);
```

求和

- 求出1到10、20到30和35到45的三个和

求和

```
int i;  
int sum;  
  
for ( i=1, sum=0; i<=10; i++ ) {  
    sum += i;  
}  
printf("%d到%d的和是%d\n", 1, 10, sum);  
  
for ( i=20, sum=0; i<=30; i++ ) {  
    sum += i;  
}  
printf("%d到%d的和是%d\n", 20, 30, sum);  
  
for ( i=35, sum=0; i<=45; i++ ) {  
    sum += i;  
}  
printf("%d到%d的和是%d\n", 35, 45, sum);
```

“代码复制”是程序质量不良的表现

- 三段几乎一模一样的代码！

求和函数

```
int i;
int sum;

for ( i=1,sum=0; i<=10; i++ ) {
    sum += i;
}
printf("%d到%d的和是%d\n", 1, 10, sum);
for ( i=20,sum=0; i<=30; i++ ) {
    sum += i;
}
printf("%d到%d的和是%d\n", 20, 30, sum);
for ( i=35,sum=0; i<=45; i++ ) {
    sum += i;
}
printf("%d到%d的和是%d\n", 35, 45, sum);
```

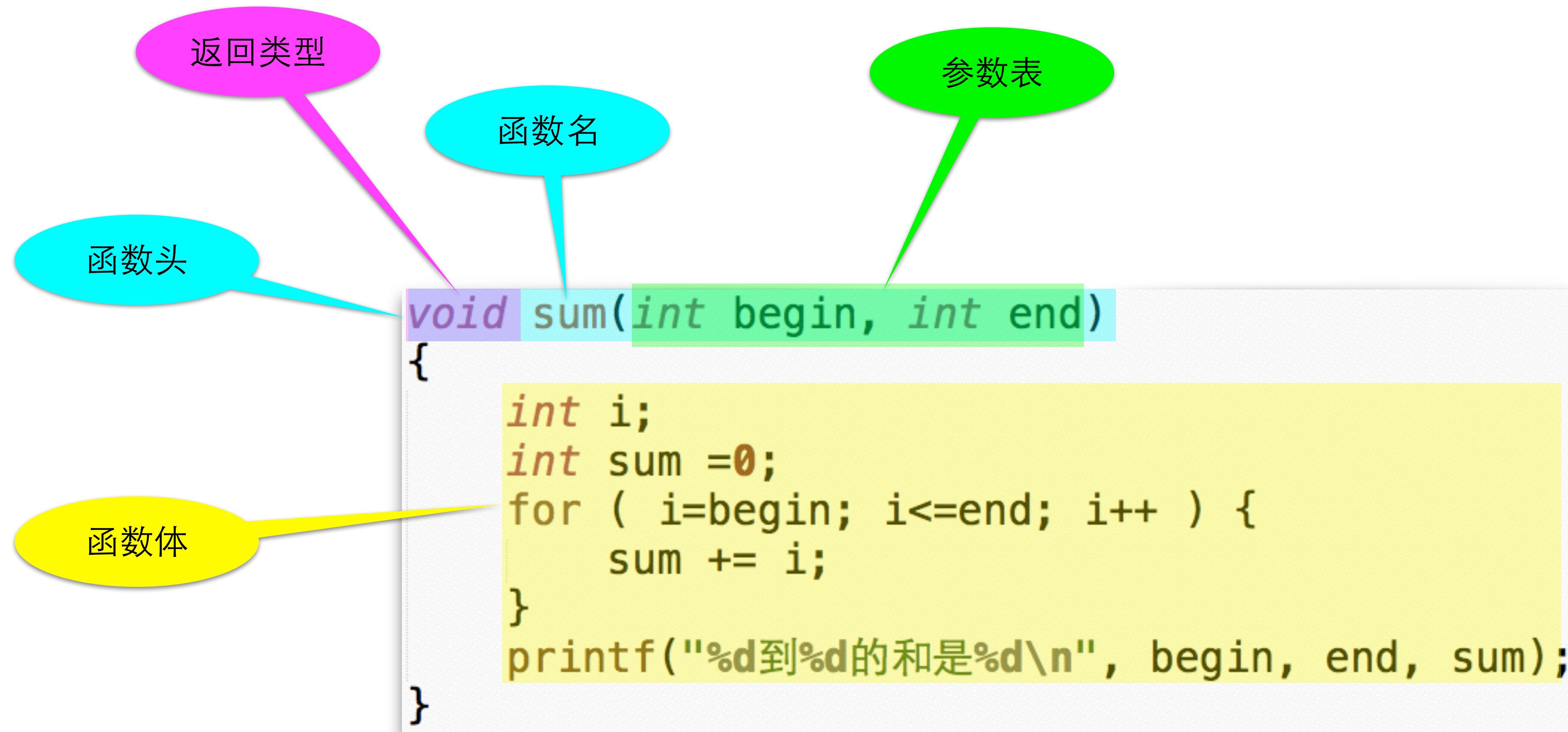
```
void sum(int begin, int end)
{
    int i;
    int sum =0;
    for ( i=begin; i<=end; i++ ) {
        sum += i;
    }
    printf("%d到%d的和是%d\n", begin, end, sum);
}

int main()
{
    sum(1,10);
    sum(20,30);
    sum(35,45);
    return 0;
}
```

什么是函数？

- 函数是一块代码，接收零个或多个参数，做一件事情，并返回零个或一个值
- 可以先想像成数学中的函数：
 - $y = f(x)$

函数定义



调用函数

- 函数名(参数值);
- ()起到了表示函数调用的重要作用
- 即使没有参数也需要()
- 如果有参数，则需要给出正确的数量和顺序
- 这些值会被按照顺序依次用来初始化函数中的参数

```
sum(1,10);
sum(20,30);
sum(35,45);
```

```
void sum(int begin, int end)
{
    int i;
    int sum =0;
    for ( i=begin; i<=end; i++ ) {
        sum += i;
    }
    printf("%d到%d的和是%d\n", begin, end, sum);
}
```

函数返回

- 函数知道每一次是哪里调用它，会返回到正确的地方

```
void sum(int begin, int end)
{
    int i;
    int sum =0;
    for ( i=begin; i<=end; i++ ) {
        sum += i;
    }
    printf("%d到%d的和是%d\n", begin, end, sum);
}
```

```
sum(1,10);
sum(20,30);
sum(35,45);

return 0;
```

从函数中返回值

```
int max(int a, int b)
{
    int ret;
    if ( a>b ) {
        ret = a;
    } else {
        ret = b;
    }

    return ret;
}
```

- **return**停止函数的执行，并送回一个值
 - **return;**
 - **return 表达式;**
 - 一个函数里可以出现多个**return**语句

从函数中返回值

```
int max(int a, int b)
{
    int ret;
    if ( a>b ) {
        ret = a;
    } else {
        ret = b;
    }
    return ret;
}
```

```
int a,b,c;
a = 5;
b = 6;
c = max(10,12);
c = max(a,b);
c = max(c, 23);
c = max(max(c,a), 5);
printf("%d\n", max(a,b));
max(12,13);
```

- 可以赋值给变量
- 可以再传递给函数
- 甚至可以丢弃
- 有的时候要的是副作用

没有返回值的函数

- void 函数名(参数表)
- 不能使用带值的return
 - 可以没有return
 - 调用的时候不能做返回值的赋值

```
void sum(int begin, int end)
{
    int i;
    int sum = 0;
    for ( i=begin; i<=end; i++ ) {
        sum += i;
    }
    printf("%d到%d的和是%d\n", begin, end, sum);
}
```