

Function

王慧妍

why@nju.edu.cn

南京大学



计算机科学与技术系



计算机软件研究所



函数

- 我们用过的

- `int printf(const char *__restrict__ _Format, ...)`
- `int scanf(const char *__restrict__ _Format, ...)`

- 封装部分功能，对外提供调用方式，便于反复使用

```
C 0hello.c > ...  
1  #include <stdio.h>  
2  
3  int main(){  
4      printf("Hello world!\n");  
5      return 0;  
6  }
```

同样的需求反复出现

- 求计算10-20,30-40,55-65的和, 分别输出
 - [sum_func.c](#)

Prime

- [prime_func.c](#)

```
#include<stdio.h>

int main(){
    int max = 100;
    //scanf("%d", &max);
    for (int i = 2; i<max; i++){
        int is_prime = 1;
        int num = i;
        for (int j = 2; j < num; j++){
            if (num % j == 0){
                is_prime = 0;
                break;
            }
        }
        if (is_prime)
            printf("%d ", i);
    }
    return 0;
}
```

函数定义与使用

每个形式参数必须有类型
如: `int a`, `int b`
可以没有形式参数 (有不建议过多)

返回类型不能是数组

不可省略返回值 (C99)
除非函数返回类型为 `void`
`void f(...){...}`

返回类型

形式参数

```
int is_Prime(int num){  
    .....  
    return res;  
}
```

函数头

函数体

可以有多条 `return` 语句,
但每次只可能执行一条,
标志函数执行结束

返回值

实际参数

```
int is_prime = is_Prime(i);
```

有返回值可按需赋值
`is_Prime(i);`

带参函数需要传入实际参数
`f();` // 不可省略 `()`

函数声明

- 在定义前需要使用函数时，提前写上函数声明
 - 在调用一个函数之前，必须对其进行声明或定义

```
int is_Prime();
```

```
int is_Prime( int );
```

```
int is_Prime( int num );
```



```
int is_Prime( int num ){  
    .....  
    return res;  
}
```

改写简单程序为带函数版本

- [numOfDigit_func.c](#)
- [leap_func.c](#)
- [stars_func.c](#)
- [sum_array_func.c](#)
- [palindrome_func.c](#)



本地变量

- 本地变量定义在块内
 - 可以定义在函数块内
 - 可以定义在语句块内
 - 也可以是随便一对大括号块内
- 变量的使用
 - 进入这个块前，其中的变量不存在，离开了这个块，其中变量消失
 - 块外定义的变量在内部依然有效
 - 块内外变量重名，则块内掩盖块外
 - 不能在一个块中定义同名变量

函数调用和参数传递

- 参数在函数调用时，会以值传递方式进行初始化
 - 形参包含的是实参的副本

```
int is_prime = is_Prime( i );
```

```
int is_Prime( int num ){  
    .....  
    return res;  
}
```

- 怎么理解？
 - Swap（有效版本 vs 无效版本）
 - 调用时候传过去的是什么？

一些值得注意的地方

- 调用时需要匹配参数类型和参数数目
 - 不匹配怎么办?
 - 编译器悄悄转换 (田螺姑娘?)
 - 已遇到原型: 根据原型要求转换类型
 - 未遇到原型: 默认实参提升 (float转double, char/short转int)

```
#include<stdio.h>
int square(int);
int main(){

    double x =3.0;
    printf("%d", square(x));
    return 0;
}
int square(int n){
    return n * n;
}
```

一些值得注意的地方

- `f(a, b) == f((a, b))//???`
 - 逗号运算符(,)是C语言运算符中优先级最低的一种运算符, 结合顺序是从左至右, 用来顺序求值
 - (最后一个逗号后面表达式的值作为整个表达式的值)
- `f(void) == f()//???`
 - 函数原型能够描述并帮助验证函数定义
- `return X; //why 0 in main?`
 - 非void需要写return, void可写可不写
 - 返回值也会隐式转换

数组作为参数的传递

- 实际上还是遵循值传递
 - 修改下试试?

```
void BubbleSort(int [], int);
```

```
void BubbleSort(int num[], int len);
```

- 数组长度如何获取?

```
int Len(int a[]){  
    int len = sizeof(a)/sizeof(a[0]);  
    return len;  
}
```

试试?
结果对么?
为什么?

数组作为参数的传递

- 数组排序
- 如何返回数组?
- `const`关键字标记变量只读

改写数组相关程序为带函数版本

- [mergesort_func.c](#)
- [bubblesort_func.c](#)
- [selectsort_func.c](#)
- [chess_func.c](#)



函数返回值

- return
 - 返回到函数被调用的地方
- exit
 - 退出整个程序运行
 - main退出

函数运行细节

- 利用堆栈实现函数的调用与返回
 - Programs run in memory (内存; 記憶體).
 - Memory = Stack (栈区) + Heap (堆区) + ...
 - Each function call has its own stack frame (栈帧).
 - Stack grows/shrinks with function calls and returns.
- [Visualization of Function Calls @ C Tutor](#)

End
