# 程序设计及算法语言

## 第三章 函数

戚隆宁
Email: longn_qi@seu.edu.cn
Tel: 13813839703

# 本章目录

# 函数的应用背景

- 代码模块化：程序需要有清晰的层次结构，提高设计的可理解性和可维护性。
- 模块参数化：相同结构代码的抽象，提高设计效率。

# 函数的概念

- 函数（function）的概念
  - 数据处理过程的独立参数化抽象表达
    - 独立的名字：函数名
    - 独立的代码地址空间：函数入口地址（entry）
    - 独立的数据地址空间：函数栈（call stack）
    - 参数化的输入数据：函数的形式参数（argument）
    - 类型化的处理结果：函数的返回值（return value）

# 函数的定义

- 函数的定义要素
  - 函数名
    - 遵从标识符定义规则
  - 形式参数
    - 定义输入数据的类型、数量、输入顺序和在函数内的标识符
  - 返回值类型
    - 定义输出数据的类型
  - 函数体
    - 定义了数据的处理过程，**return**语句返回处理结果

定义语法：

返回值类型 函数名 ( 参数列表 ) { 函数体 }

```
int SquareSum ( int a, int b )
{
    int result;
    result = a * a + b * b;
    return result;
}
```

# 函数的定义

- 允许无参数列表：void
  - 缺省为void
- 允许无返回值：void
  - 缺省为int
- 允许函数体为空：空函数
- 只允许定义一次

```
int abs( int x )
{ return x > 0 ? x : -x; }

void show( int x, int y )
{
    cout<<x<<' '<<y<<endl;
    return;
}

double PI( void )
{ return 3.1415926; }
```

# 函数的使用

- 函数的调用（function calling）
  - 调用者（caller）和被调用者（callee）
  - 函数名：必须与函数定义一致
  - 实际参数（parameter）：类型和输入顺序必须与函数定义可匹配，可以是常量、变量和表达式
  - 返回结果：一般应通过赋值语句保存
- 使用前必须有函数声明

函数调用语法:

函数名 ( 实际参数列表 )

例：

```
void main()
{
  int a = -4, b;
  b = abs( a );
  show( a, b );
  double f;
  f = 2 * PI( ) * abs( a-1 );
}
```

# 函数的声明

- 函数原型声明（function declaration /function prototype）
  - 对函数名、参数和返回值类型的定义必须与声明一致，但参数名不必一致（允许不写）。
  - 定义的同时即声明。
  - 允许重复声明，但要保持一致。
  - 一般在函数外声明。
- 声明语法

返回类型 函数名 ( 参数类型列表 ) ;

例：

```
int abs( int x ) ;
void show( int, int );
double PI( void ) ;
```

# 函数的参数传递方式

- 值传递：类型一致，自动提升或转换
- 引用传递：传递实参地址，注意常量限制
- 缺省传递（默认参数）：实参预先指定（一般声明时指定）
  - 缺省参数只允许从最右向左连续设置
  - 缺省值是调用前可确定的值，如常量、全局变量、表达式和函数返回值等

# 局部变量

- 局部变量（local variable ）的概念
  - 在局部作用域（简称局部域）定义的变量，变量名仅局部域内变量定义之后区域可见

**作用域（scope）：**

（变量和函数）标识符可被使用的范围

**局部域（local scope）**

- 函数的形参列表和整个函数体
- if、else、switch的判断条件和候选分支
- while循环的条件判断、for循环的循环控制语句和循环体
- 自定义复合语句块:{}

# 局部变量

- 局部变量（local variable ）的性质
  - 作用域：局部域{}
  - 生命期：局部生命期，从定义时开始，到所在局部域结束
  - 存储区域：函数的栈中（有的会被优化存储在寄存器里）
  - 未初始化的初始值：随机

# 全局变量

- 全局变量（global variable）
  - 作用域：全局域
  - 生命期：静态生命期，程序运行开始，到程序运行结束
  - 存储区域：程序的数据区
  - 未初始化的初始值：0

**全局域（global scope）**

- 范围从标识符定义/声明处到当前文件结束
- 在其他文件中访问时需要进行外部声明（extern）

# 局部静态变量

- 局部静态变量（local static variable）：
  - 作用域：局部域
  - 生命期：静态生命期，程序运行开始，到程序运行结束
  - 存储区域：程序的数据区
  - 未初始化的初始值：0

# 同名隐藏

- 作用域嵌套：内层域、外层域
- 同名隐藏（hide）：
  - 内层域只识别本层定义或声明的标识符，隐藏外层域同名标识符
  - 可以通过域引用符（::）解除隐藏访问（某个名字空间的）全局域或类域定义的标识符

# 函数的递归调用

- 函数的递归调用（recursion）
  - 对函数自身的直接或间接调用（直接/间接递归）
  - 在调用前必须有递归中止条件的判断

# 库函数

- 数学库函数
- 字符/字符串库函数
- 输入输出库函数
- 其他库函数

在线参考资料

http://www.cplusplus.com/reference/

https://zh.cppreference.com/w/%E9%A6%96%E9%A1%B5

# 数学库函数

- C标准数学库：<cmath>或<math.h>
  - 三角函数：cos/sin/tan/acos/asin/atan…
  - 指数、幂函数：exp/pow/sqrt…
  - 对数函数：log/log10…
  - 其他：abs/fabs/ceil/floor…
- 复数运算库：<complex>
- 数值矢量运算库：<valarray>
- 数值序列运算库：<numeric>

# 字符/字符串库函数

- 字符处理库：<cstdlib>
  - 字符处理函数：isalpha、isdigit、isupper、islower、tolower、toupper、atoi、atof、itoa
- 字符串处理库：<cstring>或<string.h>
  - 字符串长度：strlen
  - 字符串复制：strcpy
  - 字符串查找：strchr/strstr/strtok
  - 字符串比较：strcmp
  - 字符串连接：strcat

# 输入输出库函数

- 标准输入输出
- 非标准输入输出
- 文件输入输出

# 标准输入输出

- 流类库： <iostream>
- 全局文本流对象
  - 标准缓冲输出流：cout
    - 格式控制函数：setw(int)、 setfill(char)、 setbase(int)、 setprecision(int)
  - 标准缓冲输入流：cin（换行符为输入缓冲结束符）
    - 提取运算符：>> 空白字符为分隔符（提取时默认忽略，不忽略 noskipws ）
    - 状态判断函数：eof(), bad(), fail(), good()
    - 状态判断运算符： !
    - 状态恢复函数：clear(int)
    - 字符提取函数：get()（返回整型ASCII码）、get(char &)、ignore(int=1,int=EOF)
    - 字符串提取函数：get(char *, int, char='\n')、getline(char *, int, char='\n')

# C标准输入输出

- C标准输入输出库函数<stdio.h>
  - 输出函数：printf/putc/putchar/puts
  - 输入函数：scanf/getc/getchar/gets

# 非标准输入输出

- 控制台输入输出函数<conio.h>
  - 无缓冲输入函数：_getch()、_getche()
  - 按键敲击判断函数：_kbhit()

# 文件输入输出

- 文件流类库：<fstream>
- 文件流: ifstream, ofstream
- 文件操作
  - 打开文件：open(char *, int =ios::in, int = filebuf::openprot)
  - 读写文件： >> 和 <<
  - 关闭文件：close()
  - 流状态判断：is_open()和标准流状态函数
- C文件输入输出函数库<stdio.h>
  - fopen、fclose、fprintf、fscanf、fread、fwrite

# 其他库函数

- C标准库函数<cstdlib>
  - 随机数函数：rand、srand
  - 系统命令调用函数：system
- C标准日期时间库函数<ctime>
  - 日期时间函数：time 、clock
- OS相关函数<windows.h>、<unistd.h>
  - 延时等待函数：Sleep(ms)、sleep(s)/usleep(us)

# 函数重载

- 重载（overloading）函数
  - 同一作用域内
  - 函数名相同
  - 参数列表不同
- 重载匹配（实参匹配形参）
  - 优先顺序：精确匹配>类型提升/类型转换
- 二义性（ambiguous）
  - 存在多个同等可匹配函数

# 内联函数

- 内联（inline）函数
  - inline修饰的函数定义
  - 空间换时间：代码嵌入代替函数调用
  - 提高可读性
  - 仅仅是建议内联，不是必然内联
  - 内联原则：短小而频繁使用的代码
  - 包含switch、复杂的if else嵌套和while语句的函数以及递归函数等，无法内联，实现为本地全局域的函数

# 多文件结构

- 多文件结构
  - 模块化、层次化，便于大项目的设计和维护
  - 外部全局（外部变量/函数）：extern修饰（全局标识符默认修饰）
  - 本地静态（静态变量/函数）：static修饰（静态生命期同全局变量）
- 程序构建
  - 源文件（.cpp）：仅在本地使用的自定义类型和宏、全局静态变量定义、函数定义、包含所需头文件（系统的<>和本地优先的“”）
  - 头文件（.h）：给其他源文件使用的自定义类型和宏、外部变量和函数的声明以及内联函数的定义，通过编译预处理与源文件融合
  - 编译目标文件（.obj）
  - 静态库文件（.lib）
  - 可执行文件（.exe/.dll）

# 编译预处理

- 编译预处理指令
  - 宏定义指令
    - #define 宏名称 宏内容
    - #define 宏名称( 宏参数表 ) 宏内容
    - 本质是文字替换，可展开
    - 应用：固定/可设置的常量定义，增加可读性，提高修改效率
  - 文件包含指令
    - #include <文件名>
      - 只根据编译器选项设置的搜索目录搜索
    - #include “文件名”
      - 优先搜索源代码所在的当前目录
  - 条件编译指令
    - #if、#ifdef、#else、#elsif、#endif