

C程序设计语言

第4章 函数与程序结构

孙志岗
sun@hit.edu.cn
<http://sunner.cn>

大话三国

懿问曰：“孔明寝食及事之烦简若何？”

使者曰：“丞相夙兴夜寐，罚二十以上皆亲览焉。所啖之食，日不过数升。”

懿顾谓诸将曰：“孔明食少事烦，其能久乎？”

程序设计的艺术

- 算法设计的艺术
 - 程序的灵魂
 - Donald E. Knuth, "The Art of Computer Programming", 清华大学出版社, 2002
- 结构设计的艺术
 - 程序的肉体
 - *"..the larger the project, the more essential the structuring!"*—Dijkstra, 1968
 - 模块化 (Parnas, 1972)
 - 结构化 (Structural)
 - 面向对象 (Object-Oriented)
 - 面向组件 (Component-Oriented)
 - 面向智能体 (Agent-Oriented)
 -

假如不模块化

- 读多少行的程序能让你不头疼？
- `main()` 当中能放多少行程序？
- 假如 `printf()` 函数由 10 行代码替换，那么你见过的程序会成什么样子？
- 如果所有代码都在 `main()` 当中，怎么团队合作？
- 如果代码都在一个文件中，怎么团队合作？

模块化的优点

- 模块各司其职
 - 每个模块只负责一件事情，它可以更专心
 - 便于进行单个模块的设计、开发、调试、测试和维护等工作
 - 一个模块一个模块地完成，最后再将它们集成
- 开发人员各司其职
 - 按模块分配任务，职责明确
 - 并行开发，缩短开发时间
- 分而治之（Wirth, 1971）
信息隐藏（Parnas, 1972）

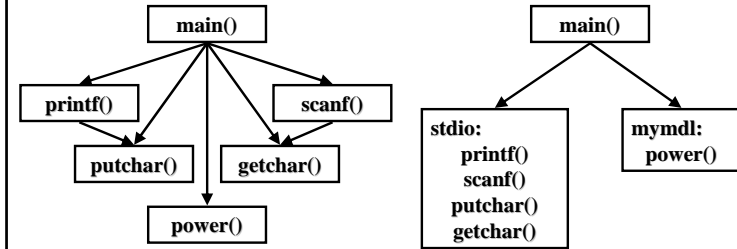
2004-12-19

Functions and Program Structure

5

函数（function）和模块（module）

- 函数是C语言中模块化编程的最小单位
 - 可以把每个函数看作一个模块
- 若干相关的函数可以合并作一个“模块”



2004-12-19

Functions and Program Structure

6

函数的分类

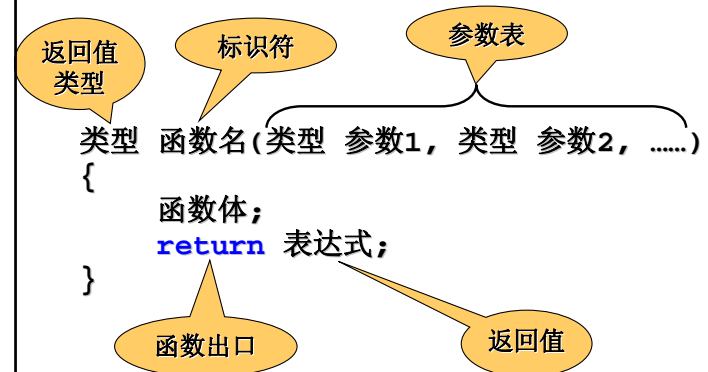
- 函数生来都是平等的，没有高低贵贱之分，只有main()稍微特殊一点点
- 库函数
 - ANSI C定义的标准库函数
 - 符合标准的C语言编译器必须提供这些函数
 - 函数的行为也要符合ANSI C的定义
 - 第三方库函数
 - 由其它厂商自行开发的C语言函数库
 - 不在标准范围内，能扩充C语言的功能
- 自定义函数
 - 自己编写的函数
 - 包装后，也可成为函数库，供别人使用

2004-12-19

Functions and Program Structure

7

函数定义（definition）



2004-12-19

Functions and Program Structure

8

函数定义 (definition)

- 函数是这样的一种运算:
 - 函数名说明运算规则
 - 参数是运算的操作数
 - 返回值是运算的结果
- 当函数执行到return语句或}时, 函数的运算停止。程序从当次调用函数的地方继续执行
 - 函数可以有多个return, 但最好只有一个且是最后一行
- 用void定义返回值类型
 - 函数没有运算结果, 没有返回值
 - return语句之后不需要任何表达式
- 用void定义参数, 表示没有参数
- 参数表里的参数 (叫形式参数, parameter) 也是函数的语句块内的变量

2004-12-19

Functions and Program Structure

9

函数调用 (call)

- 函数名(表达式1, 表达式2,);
- 调用一个函数之前, 先要对其返回值类型、函数名和参数进行声明 (declare)
 - 不对函数进行声明是非常危险的
 - 函数定义也有声明函数的效果
- 调用函数时, 提供的表达式 (叫实际参数, argument) 和该函数的形式参数必须匹配
 - 数目一致
 - 类型一一对应 (会发生自动类型转换)
 - 表达式的值赋值给对应的参数
- 返回值可以按需处理

realeq1.c

2004-12-19

Functions and Program Structure

10

函数调用的过程

- 函数的每次执行都会建立一个全新的独立的环境
 - 在“栈”中为函数的每个变量 (包括形式参数) 分配内存
 - 把实际参数的值复制给形式参数
 - 开始执行函数内的第一条语句
- 函数内的代码在这个独立的环境内工作
- 函数退出时
 - 求出返回值, 将其存入一个可以被调用者访问的地方 (x86中通常使用EAX寄存器)
 - 收回分配给所有变量 (包括形式参数) 的内存
 - 程序控制权交给调用者, 调用者拿到返回值, 将其作为函数调用表达式的结果

2004-12-19

Functions and Program Structure

11

main()、printf()和scanf() 特殊吗?

- main()
 - C语言允许不对函数参数和返回值类型进行说明
 - 甚至可以连函数名都不声明
 - 此时默认
 - 该函数的参数是不定个数的int型
 - 该函数返回值为int型
 - 永远不要利用此特性!
- printf()、scanf()
 - 变长参数表, <stdarg.h>
 - 缺点: 对参数类型和个数无法严格验证, 易使用出错

2004-12-19

Functions and Program Structure

12

使用函数要注意

- 每个函数只完成一个功能（包括main()）
 - 对函数的功能可以用不含连词的一句话描述
- 函数不能过长
 - 1986年IBM在OS/360的研究结果：大多数有错误的函数都大于500行
 - 1991年对148,000行代码的研究表明：小于143行的函数比更长的函数更容易维护
- 函数一定要对传进来的非法参数做点什么
 - 向调用者提供错误信息
 - `assert()`

safediv.c

2004-12-19

Functions and Program Structure

13

用函数完成此题

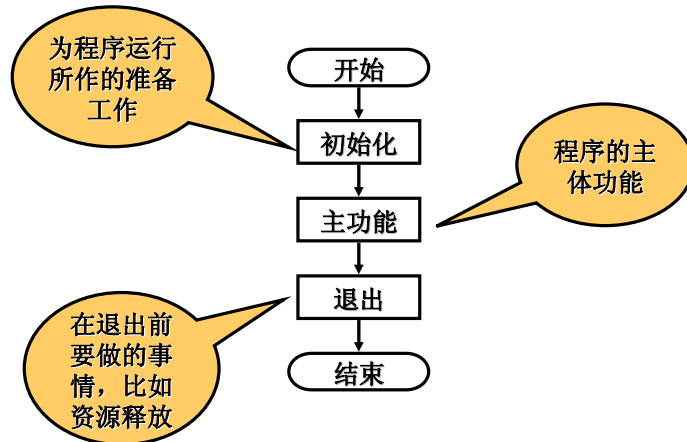
- 编程先由计算机“想”一个1到100之间的数请人猜，如果人猜对了，则计算机给出提示：“Right!”，否则提示：“Wrong!”，并告诉人所猜的数是大(Too high)还是小(Too low)，最多可以猜10次。如果猜了10次仍未猜中的话，则停止本次猜数，然后继续猜下一个数。每次运行程序可以反复猜多个数，直到操作者想停止时才结束。

2004-12-19

Functions and Program Structure

14

框架流程

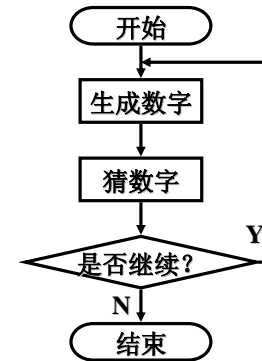


2004-12-19

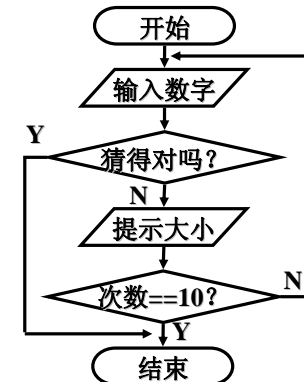
Functions and Program Structure

15

主功能



猜数字



2004-12-19

Functions and Program Structure

16

全局变量 (Global Variable)

- 在所有函数之外定义的变量是全局变量，在定义它的位置以后都有效
 - 全局变量自动初始化为0
 - 全局变量使函数之间的数据交换更容易，效率也高一些
 - 但是不推荐使用，甚至禁止使用
- 程序的任何部分都可以改写全局变量，很难确定在程序的哪里改写了它，程序结构混乱
- 不得不用时的时候（这种情况比较少见），要严格控制在它的改写

2004-12-19

Functions and Program Structure

17

静态变量 (static)

- 函数的内部变量在函数退出后失效（内存释放）。再次进入函数，变量重新定义
 - 每次函数执行都建立一个全新的执行环境，不受其它函数的干扰
- 把此变量定义为**static**，则变量的值可以保存到下次进入函数
 - `static int i;`
- 静态变量自动初始化为0

static.c

2004-12-19

Functions and Program Structure

18

寄存器变量 (register)

- 寄存器是CPU内的高速但容量有限的存储器？
- 热点变量声明为寄存器变量，访问速度更快
- `register int i;`
- 现代编译器有能力自动把普通变量优化为寄存器变量，并且可以忽略用户的指定，所以一般无需特别说明变量为寄存器变量

2004-12-19

Functions and Program Structure

19

递归 (Recursion)

- 函数直接或间接调用自己为递归
- `unsigned int func(unsigned int n)`

```
{
    if (n == 0)
        return 1;
    else
        return n * func(n-1);
}
```

recur.c

2004-12-19

Functions and Program Structure

20

模块

- 模块包含两部分
 - 源文件(`xxx.c`): 一系列相关函数的定义
 - 头文件(`xxx.h`): 这些函数的声明等必要信息
 - 函数声明、外部变量声明、宏定义、类型定义.....
- 可以将模块编译为`.obj`文件, 同`.h`文件一起供别人使用, 从而保护了源代码
- 使用模块的过程
 1. 建立一个工程 (project)
 2. 把各模块都加入到工程中
 3. `#include`模块的头文件
 4. 开始使用此模块

2004-12-19

Functions and Program Structure

21

编写模块的技术

- 模块的信息隐藏
 - 用`static`定义的函数和全局变量只在此模块内有效 (建议采用)
- 允许被其它模块使用的全局变量
 - 在源文件中定义, 不加`static`修饰
 - 在头文件中进行声明, 加`extern`修饰

2004-12-19

Functions and Program Structure

22

预编译指令

- 编译器在开始正式编译之前处理的指令, 叫预编译指令
 - 它们不会存在于最后生成的目标代码中
- 文件包含: `#include`
 - 用`#include`指定的文件内容替换`#include`所在的行
 - 用`<>`或者`" "`括上文件名
 - `<>`表示在编译器的`include`目录内查找文件
 - `" "`表示在当前目录查找文件
 - 文件名中可以带有路径

2004-12-19

Functions and Program Structure

23

`#define`

- `#define` 宏名字 替换文本
- 在`#define`之后, 所有独立出现“宏名字”的地方 (除了字符串内) 都被“替换文本”替换
- “替换文本”中可以有空格 宏名中间不要有空格
- 宏可以有参数
 - `#define max(A,B) ((A) > (B) ? (A) : (B))`
 - 能想出带参数的宏和函数的区别吗?
- 定义宏的时候注意替换发生后产生的非预想结果
 - 一般用括号可以避免, 如上例

2004-12-19

Functions and Program Structure

24

与#define配套者

- #undef, 从现在开始取消#define的定义
 - #undef MAXLINE
- #if, #else, #elif, #endif
- #ifdef, #ifndef
- 这些预编译指令通常用来处理多文件工程和程序多版本的问题。(程序多版本一般是不同平台的版本, 不同用户等级的版本, 不同开发阶段的版本等)

使用预编译指令的目的

- 增强程序可读性
 - 但是调错时宏可能带来很多难题
- 精简源代码, 提取变化
 - 这一点更多时候用函数的效果更好, 但宏也有其不可替代的优势
- 不编译无用代码, 精炼目标代码