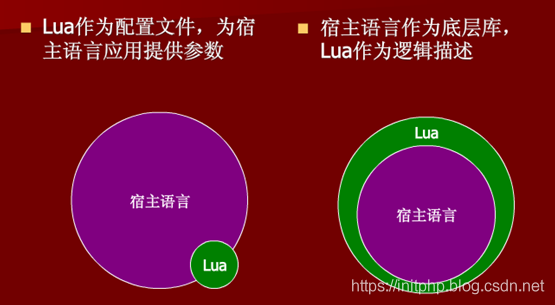
## 一、Lua简介

lua本身是一门用c语言编写的脚本语言，可用来做web脚本、游戏脚本等

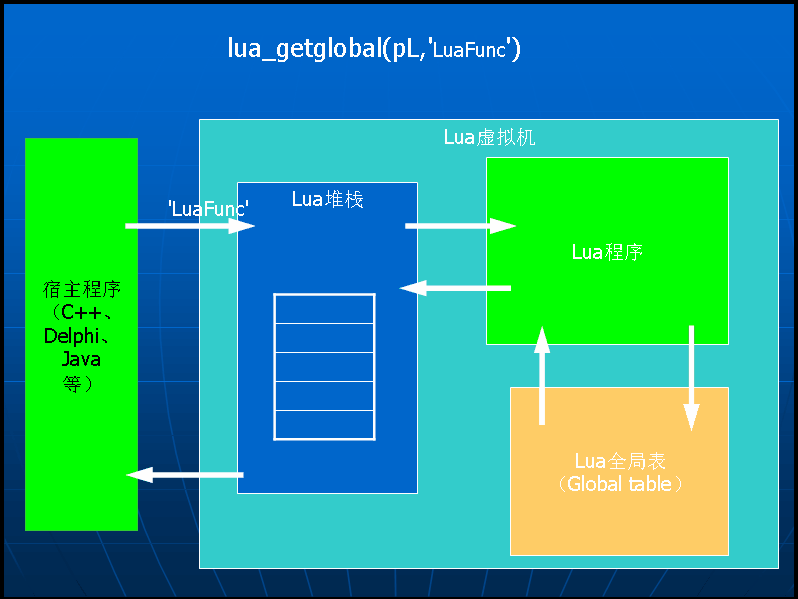
Lua语言本身的开发有两种：

1. 作为配置文件，为宿主语言提供参数
2. 作为脚本语言嵌入到宿主语言中，宿主语言作为一个库提供相关函数供lua调用

而我们的cocos2d-x引擎则是作为一个库提供函数给lua调用，在lua中写具体的逻辑实现。



下图为lua与c++的交互流程



## 二、Lua、luajit、tolua++等的区别

Lua：就是最原始的lua语言，lua源码包括如下内容



主要核心功能分三块：lua虚拟机、lua编译器、内嵌库

虚拟机作为lua的执行入口，lua编译器负责解析lua语意，内嵌库提供需要用到的基本操作函数。

Luajit：全称是Just-In-Time Compiler (JIT) for the Lua programming language. 就是lua的一个高效版本。它采用了静态编译+动态编译的混合方式，常规lua作为脚本语言，是动态加载的，随加载随编译，这样导致对同一个重复代码段的调用也是要编译；而luajit采用了对于代码，第一次使用编译，下一次使用同一块代码直接使用其编译过的机器码而不再动态编译，因此提升了效率。

tolua++:是lua的一个扩展工具，为其它语言binding到lua提供了现成的调用形式，方便将c/c++代码封装成lua。

我们框架中采用的是luajit+tolua++的解决方案，而我本篇将讲述原生的lua与c交互。

1. lua与c交互

事例见github地址:

<https://github.com/hxw956631920/cpp2lua.git>

以下为例程目录

1. cpp读取lua中的基本数据
2. cpp调用lua中的函数
3. cpp调用lua中require别的文件
4. cpp调用lua中的元表
5. lua调用cpp的变量
6. lua调用cpp创建的表和为表设置元表
7. lua调用cpp的函数
8. lua调用以模块包装的cpp函数
9. lua调用cpp类以模块注册
10. lua调用cpp类以面向对象注册
11. Lua api

将栈中idx位置的值拷贝一份压如栈顶  
void lua\_pushvalue (lua\_State \*L, int idx);  
移除idx位置的值

void lua\_remove (lua\_State \*L, int idx);  
弹出栈顶的值，将该值插入栈中idx位置  
void lua\_insert (lua\_State \*L, int idx);  
弹出栈顶的值，将该值替换到栈中idx位置  
void lua\_replace (lua\_State \*L, int idx);  
获取栈中idx位置的值的类型

LUA\_TNIL 0 LUA\_TBOOLEAN 1 LUA\_TLIGHTUSERDATA 2 LUA\_TNUMBER 3 LUA\_TSTRING 4 LUA\_TTABLE 5 LUA\_TFUNCTION 6 LUA\_TUSERDATA 7 LUA\_TTHREAD 8

LUA\_API int (lua\_type) (lua\_State \*L, int idx);  
  
对\_G进行查找名字为name的值  
LUA\_API int (lua\_getglobal) (lua\_State \*L, const char \*name);  
对栈顶idx位置的值以栈顶的值为k进行索引找到对应的value，并将其压入栈顶 该方法会触发源方法（相对于lua\_rawget而言 具体这块还没吃透 下次更新）  
LUA\_API int (lua\_gettable) (lua\_State \*L, int idx);  
对栈中位于idx位置的值进行查找其键值为k对应的value，并将value压入至栈顶  
LUA\_API int (lua\_getfield) (lua\_State \*L, int idx, const char \*k);  
注:该函数可等价于  lua\_pushstring(L, k); lua\_gettable(L, idx);  将要查找的key先压入栈顶，再调用gettable函数获取到idx位置的值，并对该值进行查找key对应的value  
  
对位于栈顶的值在\_G中设置全局变量名 并且从栈顶弹出该值  
LUA\_API void (lua\_setglobal) (lua\_State \*L, const char \*name);  
对位于栈中idx位置的值，以栈-2位置值为value，栈顶为key作为键值对出栈设置到其中 该方法会触发源方法（相对于lua\_rawset而言 具体这块还没吃透 下次更新）  
LUA\_API void (lua\_settable) (lua\_State \*L, int idx);  
对栈中位于idx位置的值进行查找键值为k，并栈顶元素出栈赋值作为k对应的value   
LUA\_API void (lua\_setfield) (lua\_State \*L, int idx, const char \*k);  
注:该函数可等价于  lua\_pushstring(L, k);lua\_pushstring(L, value) lua\_settable(L, idx);

创建一个表置于栈顶 narr为该表预开辟的narr个array元素 nrec为预开辟的nrec个非narr元素  
LUA\_API void (lua\_createtable) (lua\_State \*L, int narr, int nrec);  
创建新表 等价于 lua\_createtable(L, 0, 0);  
lua\_newtable(L);