Język skryptowy Lua

Opis języka

O języku

Lua to język skryptowy, zaimplementowany jako niewielka biblioteka napisana w C. Główne założenia projektowe języka to szybkość, przenośność i prostota. Jest on głównie wykorzystywany w celu pisanie skryptów do gier, używają go takie produkcje jak *Far Cry*, *World of Warcraft*, czy *The Witcher*.

Obecna wersja biblioteki to 5.1.4.

Zalety

- Prosta składnia, momentami bardzo podobna do C++
- Niewielki rozmiar; wersja binarna waży zaledwie 266 kB
- Wydajność; uważany za jeden z najszybszych
- Przenośność; działa na wszystkich platformach posiadający kompilator ANSI /ISO C, zarówno na Windows jak i UNIX oraz na platformach mobilnych
- Spore możliwości;
- Wysoki poziom kodu;
- Brak sprecyzowanych typów danych; przekłada się to na uniwersalność
- Łatwa integralność z kodem natywnym na poziomie funkcji;
- Udostępniana na licencji MIT/X11; prawie nieograniczone prawo do użytkowania

Wady

- Używanie natywnych struktur w skrypcie możliwe tylko interpretując je jako tablice lub używając wrapperów, np. luabind, ToLua++;
- Brak możliwości odziedziczenia struktury;
- Brak przeładowywania funkcji;
- Brak możliwości operacji binarnych; brak operatorów OR, XOR
- Brak typów wyliczeniowych;
- Brak sprecyzowanych typów danych; łatwo o błąd
- Rozpoczynanie indeksowania tablic od „1”; nie od „0”
- Momentami uciążliwa składnia; konieczność pisania „do” po „for”, „then” po „if” itp.

Ocena

Osobiście uważam, że język ten może z powodzeniem zostać wykorzystany w naszym projekcie, biorąc pod uwagę niezbyt duży rozmiar przedsięwzięcia (oczywiście w porównaniu do komercyjnych produkcji). Lua pozwala na szybkie i wygodne pisanie skryptów, a część jego (lub jej) problemów można szybko rozwiązać. Jednak jeśli miałbym być obiektywny w ocenie, poczekałbym na testy kolegów, którzy otrzymali zadanie podobne do mojego.

Mini-tutorial Lua

Poniższy mini-tutorial stanowi tylko niewielki wstęp do programowania w Lua, w którym skupiłem się głównie na jej integracji z C++. Po więcej informacji o składni języka oraz funkcjach C biblioteki odsyłam do dokumentacji.

Przygotowanie IDE

Aby móc korzystać z biblioteki należy ściągnąć jej wersję binarną. Można ją znaleźć na oficjalnej stronie Lua. Można również ściągnąć źródła i samemu ją skompilować. Nie będę opisywał dokładnej instalacji, mam nadzieję, że każdy jest w stanie zrobić to samodzielnie.

Inicjalizacja Lua (pierwszy program)

By przygotować kod natywny do korzystania z języka skryptowego załączamy odpowiednie nagłówki. Biblioteka napisana jest w C, więc trzeba podczas załączania powiadomić o tym kompilator za pomocą słowa kluczowego `extern "C"`. Trzeba również pamiętać o podłączeniu biblioteki statycznej lua5.1.lib w opcjach projektu oraz o wrzuceniu biblioteki dynamicznej lua5.1.dll do folderu projektu lub do WINDOWS/system32.

Zaczynamy definiując wskaźnik na interpretator LUA oraz tworzymy obiekt, na który będzie pokazywał. Wskaźnika tego używa się w większości funkcji. Teraz otwieramy „biblioteki” języka. Gdy Lua nie jest nam już potrzebna, zamykamy ją.

W przykładowym programie wywoływany jest jeden skrypt z pliku.

Oto przykładowy kod:

```cpp
// main.cpp

#include <iostream>

/* Pliki naglowkowe Lua */
extern "C"
{
#include <lua.h>
#include <lualib.h>
#include <lauxlib.h>
}

/* Wskaznik na interpretator Lua */
lua_State *L;

int main(void)
{
          /* Inicjalizacja Lua */
          L = lua_open();
          /* Otwarcie bibliotek */
          luaL_openlibs(L);

          /* Wywolanie skryptu */
          luaL_dofile(L, "skrypt.lua");

          /* Zamkniecie Lua */
          lua_close(L);
}
```

```lua
-- skrypt.lua

[[-- wydrukowanie wiadomosci --]]
print("Hello, Lua!")
```

Wywoływanie funkcji ze skryptu w kodzie C++

Spójrzmy na definicję prostej funkcji Lua:

```lua
-- suma.lua

[[-- funkcja sumujaca --]]
function suma(x, y)
return x + y
end
```

By wywołać ją ze skryptu w kodzie w C++ należy stworzyć dowolną funkcję, w której wrzucimy na stos parametry funkcji Lua oraz pobierzemy wartość zwracaną. Na początku wyznaczamy funkcję, którą chcemy wywołać, następnie wrzucamy na stos parametry. Wywołujemy funkcję, pobieramy rezultat i zwalniamy go.

```
// main.cpp

#include <iostream>

/* Pliki naglowkowe Lua */
extern "C"
{
#include <lua.h>
#include <lualib.h>
#include <lauxlib.h>
}

int skrypt_suma(int x, int y);
/* Wskaznik na interpretator Lua */
lua_State *L;

int main(void)
{
          /* Inicjalizacja Lua */
          L = lua_open();
          /* Otwarcie bibliotek */
          luaL_openlibs(L);

          /* Wywolanie skryptu */
          luaL_dofile(L, "suma.lua");

          /* Wywolanie funkcji ze skryptu */
          std::cout << skrypt_suma(15, 20) << std::endl;

          /* Zamkniecie Lua */
          lua_close(L);
}

int skrypt_suma(int x, int y)
{
          int suma = 0;

          /* Wywolywana funkcja */
          lua_getglobal(L, "suma");

          /* Wrzucenie pierwszego parametru na stos */
          lua_pushnumber(L, x);
          /* Wrzucenie drugiego parametru na stos */
          lua_pushnumber(L, y);

          /* Wywolanie funkcji z dwoma argumentami i jedna zwracana
wartoscia */
          lua_call(L, 2, 1);

          /* Pobranie rezultatu ze stosu */
          suma = (int) lua_tointeger(L, -1);
          /* Zwolnienie rezultatu ze stosu */
          lua_pop(L, 1);

          return suma;
}
```

Wywoływanie funkcji C++ w skrypcie

By wywołać funkcję C++ w skrypcie należy zdefiniować funkcje o ściśle sprecyzowanej budowie. Ponieważ funkcję wywołuje skrypt, jedynym argumentem, jaki przyjmuje taka funkcja jest wskaźnik na interpretator Lua. Wartością zwracaną jest zawsze liczba zwracanych argumentów (argumentów wrzucanych na stos).

Na początku pobieramy liczbę wysłanych do funkcji argumentów. Znając liczbę argumentów możemy pobrać ich wartość. Na koniec wrzucamy na stos wynik działania funkcji i zwracamy liczbę argumentów jaką wrzuciliśmy na stos (w tym przypadku 1).

Musimy również zarejestrować naszą funkcje w kodzie, inaczej skrypt nie będzie wiedział o jej istnieniu.

```
#include <iostream>

/* Pliki naglowkowe Lua */
extern "C"
{
#include <lua.h>
#include <lualib.h>
#include <lauxlib.h>
}

static int iloczyn(lua_State *L);
/* Wskaznik na interpretator Lua */
lua_State *L;

int main(void)
{
          /* Inicjalizacja Lua */
          L = lua_open();
          /* Otwarcie bibliotek */
          luaL_openlibs(L);

          /* Rejestracja funkcji iloczyn pod nazw¹ C_iloczyn */
          lua_register(L, "C_iloczyn", iloczyn);

          /* Wywolanie skryptu */
          luaL_dofile(L, "iloczyn.lua");

          /* Zamkniecie Lua */
          lua_close(L);
}

static int iloczyn(lua_State *L)
{
          /* Pobranie liczby argumentów */
          int n = lua_gettop(L);
          double iloczyn = 0;

          for(int i = 1; i <= n; i++)
                    iloczyn = iloczyn * lua_tonumber(L, i);

          /* Wrzucenie wyniku na stos */
          lua_pushnumber(L, iloczyn);

          /* Zwrócenie liczby zwracanych argumentów */
          return 1;
}
```

```
-- iloczyn.lua

[[-- wywolanie funkcji C --]]
iloczyn = C_iloczyn(10, 10, 10)

[[-- wydrukowanie wyniku --]]
print(iloczyn)
```

Niestety łączenie Lua z C++ "ręcznie" jest dosyć uciążliwe i pracochłonne. Powstały jednak tzw. wrappery, takie jak luabind czy ToLua++, które znacznie tą czynność ułatwiają oraz dodatkowo rozszerzają możliwości Lua.

Przydatne linki

Oficjalna strona Lua
http://www.lua.org

Dokumentacja Lua
http://www.lua.org/docs.html

Oficjalna strona luabind
http://www.rasterbar.com/products/luabind.html

Oficjalna strona ToLua++
http://www.codenix.com/~tolua/