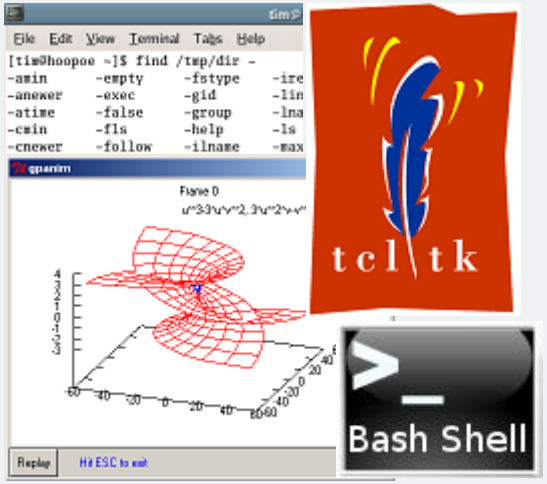
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Интерпретируемые языки программирования



Лабораторная работа №8

C++ и Lua

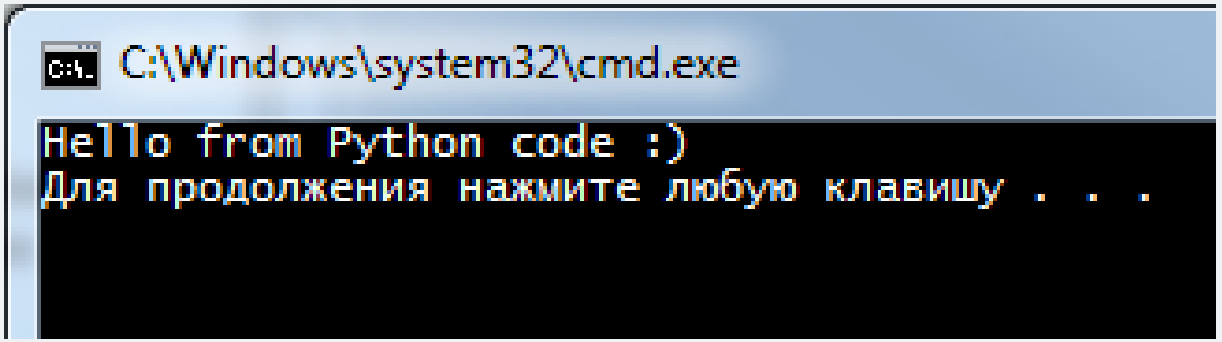
Интеграция интерпретатора Python (1)

```
#include <Python.h>
#include <iostream>
#include <cstdlib>

int main() {
  Py_Initialize();

  PyRun_SimpleString("print ('Hello from Python code :)');");

  Py_Finalize();
  return EXIT_SUCCESS;
}
```



Варианты запуска программы

```
message = "Hello World!"
print(message)
```

Вариант 1.

```
lua54.exe script.lua
```

Вариант 2.

```
luac54.exe -s -o script.out script.lua
lua54.exe script.out
```

Вариант 3.

```
lua54.exe

D:\LUA_5.4.0\bin>lua54.exe
Lua 5.4.0  Copyright (C) 1994-2020 Lua.org, PUC-Rio
> dofile("script.lua")
Hello World!
> os.exit()
```

Синтаксис языка LUA (1)

Присваивание и использование переменных:

```
-- Это однострочный комментарий
message = "Hello World!"
--[[
  Многострочный
  комментарий
--]]

print(message)

P = print
P(message)
```

```
x = "A"
y = "B"

print(x, y)
```

Параллельное присваивание

```
x, y = "A", "B"

print(x, y)
```

Синтаксис языка LUA (2)

```
num, sum = 0, 0

while num < 10 do
  if num > 5 then
    print('num is ', num)
  elseif num == 5 then
    print(num, 'is in the middle')
  end
  num = num + 1
  sum = sum + num
end

print('Sum = ', sum)

for i = 1, 10 do
  print(i)
end
```

Основные типы данных Lua

`nil`	(неопределенный)
`boolean`	(логический)
`number`	(числовой)
`string`	(строковый)
`function`	(функция)
`userdata`	(пользовательские данные)
`thread`	(поток)
`table`	(таблица)

```
D:\LUA_5.4.0\bin>lua54.exe
Lua 5.4.0  Copyright (C) 1994-2020 Lua.org, PUC-Rio
> x=5
> type(x)
number
> x='Hello!'
> type(x)
string
>
```

Использование таблиц

```
a = {1, 4, 3}
print(a[2])
```

```
options = {
  width = 10,
  height = 5
}

print(options.width)
```

```
options = {}

options.width = 10
options.height = 5

print(options.width)
```

Использование Lua в C++

```
#include <lua.hpp>
#include <LuaBridge.h>
#include <iostream>

#pragma comment(lib, "lua54.lib")

int main() {
  lua_State* L = luaL_newstate();
  luaL_openlibs(L);

  if (0 != luaL_dofile(L, "script.lua")) {
    std::cout << "__err__ : Can't open input file: script.lua" << std::endl;
    return EXIT_FAILURE;
  }

  // Тут мы пишем код
  luabridge::LuaRef n = luabridge::getGlobal(L, "test_number");

  // Конвертируем данные LUA в данные C++
  int luaNumber = n.cast<int>();

  // Выводим считанные данные на экран
  std::cout << "LUA number is '" << luaNumber << "'" << std::endl;
  return EXIT_SUCCESS;
}
```

Проверки на существование объектов

Проверка на удачную загрузку скрипта

```
if (luaL_dofile(L, "script.lua")) {
  std::cerr << "Ошибка! Не могу открыть 'script.lua'." << std::endl;
  return EXIT_FAILURE;
}
```

Проверка на удачное считывание переменных

```
if (n.isNil()) {
  std::cout << "Ошибка! Переменная 'test_number' не найдена."
            << std::endl;
  return EXIT_FAILURE;
}
```

Проверка валидности данных

```
LuaRef n = getGlobal(L, "num");
if (n.isNil()) {
  std::cout << "Ошибка! Переменная 'num' не найдена." << std::endl;
  return EXIT_FAILURE;
}

if (!n.isNumber()) {
  std::cout << "Ошибка! Переменная 'num' - не число." << std::endl;
  return EXIT_FAILURE;
}

int luaNum = n.cast<int>();
std::cout << "Number: " << luaNum << std::endl;
```

Использование таблиц

Код на LUA:

```
num = 4131

position = {
        x = 10,
        y = 20,
}
```

Код на C++:

```
LuaRef pos = getGlobal(L, "position");
LuaRef pos_x = pos["x"];
LuaRef pos_y = pos["y"];

int x = pos_x.cast<int>();
int y = pos_y.cast<int>();

std::cout << "x = " << x << std::endl;
std::cout << "y = " << y << std::endl;
```

Конфигурационный файл с настройками программы

```
window = {
  title = "Заголовок окна",
  geometry = {
    width  = 800,
    height = 600,
  }
}
```

Вызов функций Lua из C++

Код на LUA:

```
function sum_numbers(a, b)
  print("Adding "..b.." to "..a);
  return a + b
end
```

Код на C++:

```
LuaRef sum = getGlobal(L, "sum_numbers");
if (!sum.isFunction()) {
  std::cout << "Ошибка! 'sum_numbers' - не функция!" << std::endl;
  return EXIT_FAILURE;
}
std::cout << "The sum of 5 and 6 is " << sum(5, 6) << std::endl;
```

Вызов функций C++ из Lua

Код на LUA:

```
function sum_numbers(a, b)
    cpp_print("Hi there!")
    print("Adding "..b.." to "..a);
    return a + b
end
```

Код на C++:

```
void cpp_print(const std::string& s) {
  std::cout << "C++ LOG : " << s << std::endl;
}
```

```
Namespace ns = getGlobalNamespace(L);
ns.addFunction("cpp_print", cpp_print);
```