- program może tez cos prezentowac

Jak wiadomo, wiele rzeczy w dzisiejszych czasach zostało bardzo zautomatyzowane. Dotyczy to nie tylko zastąpienia ludzi robotami w miejscach pracy gdzie potrzebna jest bardzo duża precyzja produkcji, ale także do rutynowych, prostych (głównie schematycznych, które można opisać jakimiś algorytmami), aczkolwiek żmudnych prac. Automatyzacje mają także miejsce w systemie komputerowym: na przykład użytkownik może codziennie logować się na swoją pocztę czy wykonywać jakieś inne proste czynności, które musi powtarzać co ileś dni.

**- czym jest jezyk skryptowy**

Jeśli zapytamy programistę czym jest język skryptowy, zapewne w większości przypadków otrzymamy parę przykładów jaki język należy do tej kategorii. Jednak większość osób tak naprawdę nie wie dlaczego. Znaczna liczba źródeł definiuje go jako język umożliwiający sterowanie konkretnymi zadaniami w systemie (ang. Job Control)[[1]](#footnote-2). Wywodzi się on głównie z czasów superkomputera (ang. mainframe) firmy IBM®, który używał języka JCL (Job Control Language) do uruchamiania określonych programów w systemie bez interwencji manualnej (ang. batch processing[[2]](#footnote-3)). Języki te początkowo miały bardzo dużo ograniczeń, np. język JCL umożliwiał m.in. deklarację zmiennych/parametrów, warunkowe uruchamianie zadań, część możliwości modularności oraz parametryzowane procedury, które bardziej przypominały znane z C makra[[3]](#footnote-4).

Większość tego typu języków jest także określanych jako „Glue Languages”. Pojęcie to oznacza, że takie języki służą także do łączenia komponentów (narzędzi, modułów) niekoniecznie napisanych w języku skryptowym. Dzięki temu możliwe jest połączenie programów napisanych w różnych językach, które rozwiązują pojedyncze problemy, aby wspólnie rozwiązać większy problem.

Z powodu szybkiej i specyficznej ewolucji języków skryptowych (np. Perl, który posiada cechy języka skryptowego, systemowego jak i idei maszyny wirtualnej[[4]](#footnote-5) podobnej do Java™), konkretna definicja określająca czym naprawdę jest język skryptowy jest trudna do ustalenia. Cechy charakterystyczne takich języków, które są często podawane[[5]](#footnote-6) to m.in.:

- dynamiczne typowanie, np. w Lua:

x = "string"

x = 5

oba wyrażenia są poprawne i nie wymagają ręcznego określenia typu przez programistę)

- zazwyczaj są interpretowane zamiast kompilowane

- głównie używane do pisania krótszych i prostszych programów

- czym jest lua i jakie ma możliwości

Lua jest jednym z przenośnych języków skryptowych najczęściej wykorzystywanym w branży gier komputerowych[[6]](#footnote-7), który zbliżony jest najbardziej do języków proceduralnych. Działa na każdej maszynie, która udostępnia kompilator ANSI C, począwszy od systemów wbudowanych aż do mainframe’ów. Umożliwia między innymi rozszerzalność (dodawanie) funkcji, których implementacje egzystują w zewnętrznym kodzie napisanym np. w języku C[[7]](#footnote-8), Java[[8]](#footnote-9) lub Delphi[[9]](#footnote-10). Ponadto od roku 2003, gdy zadebiutowała wersja 5.0, działa na wirtualnej maszynie Parrot, która w odróżnieniu od większości wirtualnych maszyn (które są bazowane na stosie) bazuje na rejestrach[[10]](#footnote-11), dzięki temu zwiększona została, w niektórych przypadkach ponad dwukrotnie, wydajność w dostępie do zmiennych lokalnych. Co więcej posiada automatyczne zarządzanie pamięcią wraz ze swoim własnym garbage collector’em, który został dodany w wersji 5.1[[11]](#footnote-12) oraz który jest niezwykle istotny w grach komputerowych.

- jak dziala lua

Działanie interpretera Lua oparte jest o emitowanie instrukcji dla maszyny wirtualnej „w locie”, czyli w trakcie procesu parsowania, aczkolwiek zapewniając także pewne optymalizacje, np. utworzenie kodu (wyemitowanie instrukcji) dla wyrażeń związanych ze stałymi i zmiennymi jest opóźnione, tzn. interpreter najpierw generuje strukturę tzw. *tablicę asocjacyjną* (ang. associative array) - która de facto jest jedyną strukturą danych wykorzystywaną w Lua[[12]](#footnote-13) - w której umieszcza potrzebne mu dane na ich temat i na jej podstawie optymalizuje kod[[13]](#footnote-14).

- lua okiem programisty

1. RICH MORIN, VICKI BROWN, *Scripting languages*, <http://www.mactech.com/articles/mactech/Vol.15/15.09/ScriptingLanguages/index.html>, 1999 [↑](#footnote-ref-2)
2. IBM® CORPORATION, *z/OS concepts,* [http://publib.boulder.ibm.com/infocenter/ zos/basics/topic/com.ibm.zos.zconcepts/zconcepts\_book.pdf](http://publib.boulder.ibm.com/infocenter/%20zos/basics/topic/com.ibm.zos.zconcepts/zconcepts_book.pdf) str 79, 2010 [↑](#footnote-ref-3)
3. IBM® CORPORATION, *IBM System/360 Operating System Job Control Language (C28-6529-4)*, <http://www.bitsavers.org/pdf/ibm/360/os/R01-08/C28-6539-4_OS_JCL_Mar67.pdf>, 1967 [↑](#footnote-ref-4)
4. PATRICK MICHAUD, *Announce: Rakudo Star — a useful, usable, "early adopter" distribution of Perl 6*, <http://rakudo.org/2010/07/29/rakudo-star-2010-07/>, 2010 [↑](#footnote-ref-5)
5. CUNNINGHAM & CUNNINGHAM INCORPORATION, *Glue Languages,* <http://c2.com/cgi/wiki?GlueLanguages>, 2011 [↑](#footnote-ref-6)
6. GDMAG STAFF, *Front Line Award Winners – programming tools*, <http://gdmag.com/blog/2012/01/front-line-award-winners.php>, 2011 [↑](#footnote-ref-7)
7. ROBERTO IERUSALIMSCHY, *Programming in Lua, Second Edition, strona 215 (An Overview of the C API)*, 2006 [↑](#footnote-ref-8)
8. JAMES ROSEBOROUGH, IAN FARMER, *LuaJ*, <http://sourceforge.net/projects/luaj/>, 2011 [↑](#footnote-ref-9)
9. DENNIS D. SPREEN, *Lua 5.1 for Delphi* 2010 , <http://blog.spreendigital.de/2009/09/28/lua-5-1-for-delphi-2010/>, 2009 [↑](#footnote-ref-10)
10. W. CELES, H. L. FIGUEIREDO, R. IERUSALIMSCHY, *The Implementation of Lua 5.0 - The Virtual Machine*, <http://www.jucs.org/jucs_11_7/the_implementation_of_lua/jucs_11_7_1159_1176_defigueiredo.html> sekcja 7, 2005 [↑](#footnote-ref-11)
11. W. CELES, H. L. FIGUEIREDO, R. IERUSALIMSCHY, *The Implementation of Lua 5.0 - Introduction*, <http://www.jucs.org/jucs_11_7/the_implementation_of_lua/jucs_11_7_1159_1176_defigueiredo.html> sekcja 1, 2005 [↑](#footnote-ref-12)
12. W. CELES, H. L. FIGUEIREDO, R. IERUSALIMSCHY, *The Implementation of Lua 5.0 – Tables,* <http://www.jucs.org/jucs_11_7/the_implementation_of_lua/jucs_11_7_1159_1176_defigueiredo.html> sekcja 4, 2005 [↑](#footnote-ref-13)
13. W. CELES, H. L. FIGUEIREDO, R. IERUSALIMSCHY, *The Implementation of Lua 5.0 - An Overview of Lua's Design and Implementation*, <http://www.jucs.org/jucs_11_7/the_implementation_of_lua/jucs_11_7_1159_1176_defigueiredo.html> sekcja 2, 2005 [↑](#footnote-ref-14)