Introduction

Welcome to the Ring programming language!

In this chapter we are going to discuss the goals behind the language design and implementation.

Motivation

In Nov. 2011, I started to think about creating a new version of the Programming Without Coding Technology (PWCT) software from scratch.

I was interested in creating multi-platform edition of the software beside adding support for Web & Mobile development. Most of the PWCT source code was written in VFP and the software comes with a simple scripting language for creating the components called (RPWI). The software contains components that support code generation in programming languages like Harbour, C, Supernova & Python.

What i was looking for is a programming language that can be used to build the development environment, provides multi-platform support, more productivity, better performance, can be used for components scripting & can be used for developing different kinds of applications.

Instead of using a mix of programming languages, I decided to use one programming language for creating the development environment, for components scripting & for creating the applications.

I looked at many programming languages like C, C++, Java, C#, Lua, PHP, Python & Ruby. I avoided using C or C++ directly because i want high-level of productivity more than the level provided by these languages, also a language behind visual programming environment for novice programmers or professionals must be easy to use & productive.

Java & C# are avoided for some reason too! I wanted to use a dynamic programming language and these languages are static typing, Java is multi-platform, also C# through Mono, but the use of huge number of classes and forcing the use of Object-Orientation, using a verbose language is not right for me. I need a small language, but fast and productive, also I need better control on the Garbage Collector (GC), I need a better one that is designed for fast applications.

Lua is small and fast, but it’s avoided because I need more powerful language for large applications.

PHP is a Web programming language and it’s syntax is very similar to C, this leads to a language not general as I want and not simple as I need to have.

Python & Ruby are more like what I need, but I need something more simple, smaller, faster & productive.

Python and Ruby are Case-Sensitive, the list index start counting from 0, you have to define the function before calling it, Ruby usage of Object-Orientation and message passing is more than what I need and decrease performance, Python syntax (indentation, using self, :, pass & _) is not good for my goals.

All of these languages are successful languages, and very good for their domains, but what I need is a different language that comes with new ideas and intelligent implementation (Innovative, Ready, Simple, Small, Flexiable and Fast).

History

In Sept. 2013 I started the design and the implementation of the Ring programming language. After 21 months of development, In May 2015 the language Compiler & Virtual Machine were ready for use!

After that i spent three months testing the language again, trying to discover any bug to fix, writing better tests, by the end of August 2015, all know bugs were fixed, Writing many tests and testing automation using batch files and diff.exe helped a lot in getting a stable product.

In 12 September 2015, most of the documentation was written. Before releasing the language i started the marketing by writing a post in Arabic language about it to my facebook profile page asking for contributors interested in the language idea after reading a short description, in the same day i got a lot of emails from developers and friends interested to contribute!

The first version of the language Ring 1.0 is released in January 25, 2016

Features

The Ring language comes with the next features

Tip: One of the main goals behind the first release is creating a useful

language ready for production!

- Free Open Source (MIT License)
- Interpreter - Hybrid Implementation (Compiler+VM)
- Declarative programming on the top of Object-Oriented programming
- No explicit end for statements (No ; or ENTER is required)
- A small language (Around 100,000 lines of code)
    - The compiler + The Virtual Machine are 15,000 lines of C code
    - The other 85,000 lines of code are related to libraries!
        - 10,000 lines of C code
        - 50,000 lines of C++ code
        - 25,000 lines of Ring code
- Writen in ANSI C (The code is generated)
- Developed using Visual Programming (PWCT)
- Optional Printing for Tokens/Grammar/Byte-Code during execution
- Portable (Windows, Linux & Mac OS X)
- Comments (One line & Multi-lines)
- Not Case-Sensitive
- Dynamic Typing
- Weakly typed
- Lexical Scoping (Global, Local & Object State)
- Default scope for variables inside functions (Local)
- Default scope for variables outside functions (global)
- Garbage Collector - Automatic Memory Management (Escape Analysis and Reference Counting)
- Structure Programming
- Rich control structures & Operators

- For in get item by reference not value, you can read/edit the item
- Use exit to go outside from more than one loop
- Procedures/Functions
- Main Function (optional)
- Call Function before the definition
- Recursion
- Multi-line literals
- Access (read/write) string letter by index
- The list index start by 1
- No keyword to end Functions/Classes/Packages
- Range operator ex: 1:10 and “a”:”z”
- First Class Variables, Lists, Objects and Functions
- Store/Copy Lists/Objects by value (Deep Copy)
- Pass Lists/Objects by reference
- Native Object-Oriented Support
    - Encapsulation
    - Setter/Getter (optional)
    - private state (optional)
    - Instantiation
    - Polymorphism
    - Composition
    - Inheritance (Single Inheritance)
    - Operator Overloading
    - Packages
- using { } to access objects and use attributes/methods as variables/functions
- Reflection and Meta-programming
- Clear program structure (Statements then functions then packages &

classes)

- Exception Handling
- Eval() to execute code during run-time
- 8-bit clean, work on binary data directly
- I/O commands
- Math functions
- String functions
- List functions
- File processing functions
- Database support (ODBC & MySQL)
- Security Functions (OpenSSL)
- Internet Functions (LibCurl)
- CGI Library (Written in Ring)
    - HTTP Get
    - HTTP Post
    - File upload
    - Cookies
    - URL Encode
    - HTML Templates
    - HTML Special Characters
    - HTML Generation using Functions
    - HTML Generation using Classes
    - CRUD Example (using MVC)
    - Users Example (Register, Login and Check)
- Extension using C/C++ (Simple API)
- Embedding the language in C/C++ programs
- Comes with code generator (Written in Ring) to quickly wrap C/C++ Libraries
    - Used to Support Allegro by creating RingAllegro
    - Used to Support Qt by creating RingQt

- Create 2D Games (Using the Allegro Library)
- Create GUI Applications for Desktop and Mobile (Using the Qt Framework)