Technical Disclosure Commons

Defensive Publications Series

November 2020

Pass By Reference Substitution During Code Migration

Paneendra BA

Abhay Garg

Alexandre Ginet

Arijit De

Pass By Reference Substitution During Code Migration

ABSTRACT

This disclosure describes code conversion techniques for replacement of function calls that utilize pass by reference in program code in a source language with equivalent constructs in a target language that does not support pass by reference. When a pass by reference is encountered in the source program code, a wrapper object is created by wrapping a native object with a pre-defined key. In the translated code, the value of the object for the specified key is updated instead of being accessed as a normal variable. Upon return of program control from the called function, the value assigned to the object for the key is assigned back to the original variable. The variable now stores an updated value, just as it would in a pass by reference scenario.

KEYWORDS

- Pass by reference
- Visual Basic
- JavaScript
- Code migration
- Code conversion
- Legacy code
- Native type

BACKGROUND

Code written in a particular programming language may have to be converted (translated) into code in a different programming language, e.g., during migration of legacy computer applications. Some source programming languages such as Visual Basic for Applications (VBA),

support passing native type arguments to functions either by reference or by value, which can be specified in the function call. However, some target programming languages to which such code is translated, e.g. JavaScript, only support passing arguments by value and do not support passing arguments by reference. During conversion of code that includes function calls where arguments are passed by reference to such a target language, function calls need to be suitably modified using features supported by the target language.

DESCRIPTION

This disclosure describes techniques for replacement of function calls where arguments are passed by reference with equivalent function calls where arguments are passed by value. The techniques can be utilized, for example, during migration of code from one programming language that supports the passing of arguments by reference to another programming language that does not support the passing of arguments by reference. An automated tool can implement the described techniques to convert code written in a particular language that supports the passing of arguments by value or by reference, e.g., Visual Basic Applications (VBA), etc. to a different programming language such as JavaScript that only supports the passing of arguments by value for native types and passing by reference for objects/arrays.

When a pass by reference is encountered in the source program code, a wrapper object is created by wrapping a native object with a pre-defined key. In the translated code in the target language, the value of the object for the specified key is updated instead of being accessed as a normal variable. Upon return of program control from the called function, the value assigned to the object for the key is assigned back to the original variable. The variable now stores an updated value, just as it would in a pass by reference scenario.

Function definition

```
Sub Func(ByRef num As Integer,
ByVal str As String)
  num = num + 2
End Sub
```

Function call

```
Func(3, 'test')
```

(a)

Function definition

```
Function Func(numWrapper, str) {
  numWrapper[wrappedObject] =
numWrapper[wrappedObject] + 2;
}
```

Function call

```
var numWrapper = {wrappedObject: num};
Func(numWrapper, str);
num = numWrapper['wrapperObject'];
```

(b)

Fig. 1: Pass by Reference conversion; (a) original code; (b) equivalent code

Fig. 1 depicts an example function definition and function call where variables are passed by reference in source program code. Fig. 1(a) depicts code in an original programming language that utilizes pass by reference (for "num") while Fig. 1(b) depicts equivalent translated code that utilizes a pass by value mechanism ("numwrapper") to replicate the original function. As depicted in Fig. 1(b), all instances in the function definition of the object passed by reference are replaced in the translated code by the corresponding wrapper object.

Techniques of this disclosure can also be extended to arrays and properties that are passed by reference. In this case, it is ensured that the setter is called after the method invocation ends instead of trying to assign it to the variable.

In an example, the described techniques can be used to convert macros, e.g., in Microsoft Excel, that are written in VBA to JavaScript, e.g., suitable for web-based spreadsheet applications.

CONCLUSION

This disclosure describes code conversion techniques for replacement of function calls that utilize pass by reference in program code in a source language with equivalent constructs in

a target language that does not support pass by reference. When a pass by reference is encountered in the source program code, a wrapper object is created by wrapping a native object with a pre-defined key. In the translated code, the value of the object for the specified key is updated instead of being accessed as a normal variable. Upon return of program control from the called function, the value assigned to the object for the key is assigned back to the original variable. The variable now stores an updated value, just as it would in a pass by reference scenario.

REFERENCES

1. Kesavan, Raghuraman, "[Javascript] Pass By Value And Pass By Reference In JavaScript", August 2017, https://medium.com/nodesimplified/javascript-pass-by-value-and-pass-by-reference-in-javascript-fcf10305aa9c, last accessed 25 October 2020.
2. Barbour, Brian, "Passed By Reference Vs. Value In Javascript", June 2019, https://dev.to/steelvoltage/passed-by-reference-vs-value-in-javascript-2fna, last accessed 25 October 2020.
3. Rauschmayer, Axel, "Speaking JavaScript, Chapter 15. Functions", http://speakingjs.com/es5/ch15.html, last accessed 25 October 2020.
4. Gardner, Todd H., "How to Correctly Wrap a JavaScript Function", https://trackjs.com/blog/how-to-wrap-javascript-functions/ last accessed 25 October 2020.
5. VBA-to-JavaScript-Translator, https://github.com/mha105/VBA-to-JavaScript-Translator last accessed 25 October 2020.