

# An Experiment of Looping Argument in Hypertext Preprocessor in Web Based System

Danial Kafi Ahmad 1 2\*, Norshima Zainal Shah2, Hoo Yann Seong3

1Faculty of Information Technology, INTI International University, Nilai, Malaysia

2Language Centre, National Defence University of Malaysia, Kuala Lumpur, Malaysia.

3Centre for Defence Foundation Studies, National Defence University of Malaysia, Kuala Lumpur, Malaysia.

**Abstract:-** Looping were common and essential in most software based system regardless of their programming languages. As an example of a basic system, the absent of loop function, however seems possible for the program to perform required repetition automated task, but could lead to major issues in terms of system efficiency. Imagine of a particular software system without loop function, but would require to perform multiple repetitions, then it would be a system with massive lines of code which create so called “important redundancies” in within the software code itself. In this experiment, a light program of looping function of different designs and versions were developed whereby the argument or parameter of initialization and incremental function were manipulated in a manner of non-common practice.

**Keywords:-** Control Structure, Repetition, Initialization, Argument.

## I. INTRODUCTION

Control structures were very important in any system of software based nowadays. It was classified into two types of structure which are condition and looping. Condition and looping, both perform validation of condition (TRUE/FALSE value), however, the loop function would make sure that repetition of certain program were executed as according to the required condition [1]. These situation had caused the perception of the importance of the loop function in a system at a very high level, thus it is a very good idea to understand deeply how loop functions works especially in a content of so called “non-common practice”. By possess good and deep understanding of the function, therefore the tendency to design a good software based system which integrated with loop function structure is high. In this experiment, the for() loop function were deployed in the program, as to investigate more on the non-common practice of the loop function in terms of syntax and semantic. In this case, syntax would be the code grammar and semantic would be the logical flow of the program. The program were developed using Hypertext preprocessor (PHP), in which the results would be in this scripting language context. This is due to the possibilities of other scripting language or programming language that own different interpretations towards the scripts or statements, this include the syntax and semantic.

## II. METHODOLOGY

### ➤ SDLC

Software Development Lifecycle (SDLC) were vital in any software development processes or activities as it provides the essence or the guidance of how the activities of development would be executed systematically. The systematic approach is a required and common practice in Software Engineering [2]. In other any engineering discipline, it would be the same concept in a context of systematic approach whereby the design and development should be systematic and organized. This means that measurement, records and approval take place. In this experiment of small scale software development, an incremental model which comprises of Specification, Development and Validation were executed in a non-fixed sequence of activities (Non-Plan Drive) [3]. It would mean that the activities were done according to the requirement change due to a certain factor. The non-plan driven would benefits the developer by providing flexibilities and agilities during the development as well as rapid development.

### ➤ Software Process Model (Incremental)

#### Specification

#### Development Tools

PHP scripting language were used in this research experiment in order to develop the program. This is due to the nature of its reliability and functionality as well as the popularity [4]. Popularity would plays an important role in determining the usage of the language as it would reflects the size of the community of the language. For example, PHP is an open source language which mean that it provides easy access. Developer or anyone who involved in software development would be able to use the language to develop a program without any cost incur in most of the cases. In addition, large community would be able to create a variety of problems, issues as well as solution in within the PHP language context. It seems that more problems could lead to more solutions. Therefore more learning occur directly or indirectly. The XAMPP package which consist of Apache were used in this experiment. However, there were no database system were deployed in the program.

➤ *Architecture-Client Server*

Since the program were written in a form of Web Based, therefore a components of Client and Server were required to be deployed and integrated in the system. The program were developed on the server side whereby the file which consist of the scripts or code would be stored in the server folder of localhost (*htdocs*). So, whenever the user would like to view the program’s output, therefore the client (browser) were required to request the program page form the server and this were done via the Uniform Resource Locator (URL). In summary the client would request program in form of page form the server, and if the program is exist, therefore the server would return the program to the client [5][6].

➤ *Pseudocode*

Prior to the development (coding) the program’s design of flow were depicted using the pseudocode. Even though the pseudocode were in general idea, however it would be as important as the real code in order to understand the program flow or semantic, and it also were used in testing activities of Static Type since it is not executable in automated means to see the output [7].

```
BEGIN
  for (i=1;i<=10;i++)
    display i with break space (newline)
END
```

Fig 1:- Pseudocode of first version (common practice).

```
BEGIN
  for (i<=10;i++)
    display i with break space (newline)
END
```

Fig 2:- Pseudocode of second version (without both assigned initialization and end of line indicator)

```
BEGIN
  for (;i<=10;i++)
    display i with break space (newline)
END
```

Fig 3:- Pseudocode of third version (without assigned initialization with end of line indicator).

```
BEGIN
  for (;i<=10;)
    display i with break space (newline)
    i++
```

Fig 4:- Pseudocode of fourth version (without assigned initialization with end of line indicator and without increment in within the for() argument).

➤ *Development Code*

The program were developed in four versions. This is to distinguish in between the versions of loop design. The arguments in for(arguments...)of variable initialization were manipulated in this experiment in order to validate all of the

versions’ output. Below shows the basic list of function or statement that were used in the program:

- for()
- \$i
- <=
- \$i++
- echo ()
- “<br>”
- ;

Table 1 shows the basic components that were used to form a flow chart. In most software development process, flow chart would be treated as a visual of the system design that could represent or depict the flow of the system regardless of small or large scale [8]. In this research experiment, the flowchart were used in order to shows the versions of loop function in terms of their flow and sequence.



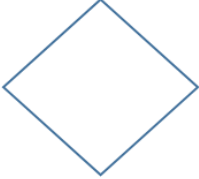

Shape	Action
	Process/Computation
	Input/Output
	Condition/Decision
	Flow/Sequence/Direction

Table 1:- Components of Flow Chart.

➤ *Installation and Configuration*

The user would need to perform a configuration of the port where necessary in order to avoid unavailable port issue. Default port would be 80 in most of the default configuration, However, there would be a situation of port were being used by other application, and this led to the necessity for port configuration and it could be done via the Graphical User Interface (GUI) of XAMPP Control Panel. In this research experiment, the port were configured to 8080 and had caused the request of the web page to be made by the client (browser) via URL in this form (localhost:8080/loop.php). Figure 5 shows the Listen 12.34.56.78:8080 and Listen:8080 indicated the changed port.

III. DISCUSSION

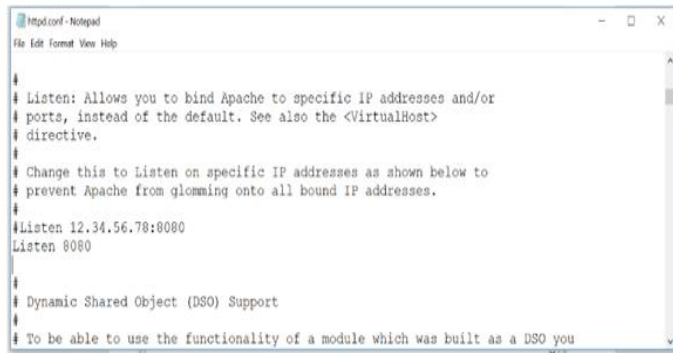


Fig 5:- Configuration of port via httpd.conf file.

➤ Validation

Testing were conducted towards all versions of the loop program in order to validate and verify the output. Following Test Level, Type and Techniques conducted as shown below:

- Unit(Functional)→Static, Dynamic
- Integration(Functional)→ Static, Dynamic
- System(Functional)→Static, Dynamic
- Acceptance(Functional)→ Static, Dynamic

➤ Test Level

The following test model were implemented to depict the testing flow of the program’s component rather than the mathematical sides [9].

1<=x<=n, 1<=y<=n,  
n=maximum number of components /execution state  
component(x) → result(y=x)

Fig 6:- Unit Testing Model.

1<=x<=n, 1<=y<=n,  
n=maximum number of components/execution state  
component(x) executed AND component(x+1) executed  
→ result(y=x)

Fig 7:- Integration Testing Model.

1<=x<=n, 1<=y<=n,  
n=maximum number of components/execution state  
(component(x) AND component(x + 1))  
result(y = x)

Fig 8:- System Testing Model.

➤ Algorithm

Coding

```
<?php
for($i=1;$i<=10;$i++){echo $i."<br>";}
?>
```

Fig 9:- Loop version 1

Figure 9 shows the code of the version 1 loop design. The design were practiced by most of the developer when implementing loop control structure, in other words it is a common design in loop programming. The code consist of few optional statements such as the curly braces ({}), and the initialization. The absence of initialization would cause to value assumption in a context of PHP. The argument in within the loop which are \$i=1; create a variable of \$i which assigned by a value of 1, and \$i<=10 is a condition which need to be met in order for the “execution statement” would be executed. However, the increment of variable’s value which represented by \$i++ were executed after the statement execution even though it is in within the for() argument and were skewed to the right position of the argument. For example as in this design and version, the initialization would take place followed by the condition validation and statement execution. The statement execution would be redirected to the for() loop again for condition checked, however, it was first that increment would occur on the variable of \$i before the condition checked for the second time. In this design, the number of 1 to 10 were displayed as an output with the line break after each of the displayed value.

```
<?php
for($i<=10;$i++){echo $i."<br>";}
?>
```

Fig 10:- Loop version 2

Figure 10 shows the loop of version 2 whereby the initialization and end of line indicator were absence. This design would produce the output with an error. In the for() loop, the initialization is technically required in within the argument, therefore when the initialization is missing, the for () loop function were affected and this cause the program to stop executing. Thus, the statement execution which displayed the number of 1-10 were not able to be displayed.

```
<?php
for(;$i<=10;$i++){echo $i."<br>";}
?>
```

Fig 11:- Loop version 3

Whilst, the loop design of version 3 as shown in figure 11 were different as compared to loop design of version 1 and version 2. The differences were the absence of initialization of integer value towards the \$i as initialization, which caused to the program to assume the value of variables to null. However in this program the null value would be less than ten in terms of numerical value or in other words would be considered as so called “zero”. In this version or design, the PHP interpreted the program as no error but notice instead due to undefined variables. Since in a single iteration, there were three variables which are \$i in \$i<=10;, \$i in echo \$i.“<br>” and \$i in \$i++, then three notices (Undefined Variable) were produced respectively as the output. The program however was able to displayed the integer of 1 until 10 as the program does not have any error since the end of line which is semicolon (;), were indicated that there is an argument before the argument of condition (\$i<=10;) in the for () loop, and the program would assume that it would be or maybe the initialization, and therefore it continues to execute with the null value or so called “zero” into the statement execution (echo \$i. “<br>”;) as the condition (\$i<=10;) is TRUE and followed by the incremental (\$i++) of the variable. The incremented value of variable \$i which is now 1 is being checked via the condition for the second iteration and displayed the value. This execution were keep until the final iteration which is the eleventh iteration followed by the loop termination.

```
<?php
for(;$i<=10;){ echo $i."<br>"; $i++;}
?>
```

Fig 12:- Loop version 4

Figure 12 shows that the design of loop without initialization of variable \$i which had caused notices produced as part of the program output, and this were similar as compared to version 3. However in version 4 as shown in figure 12, the increment of the variable \$i was written in within the “statement execution” which belongs to for() loop. This design would produce similar output as to version 3, as the increment would happen after echo \$i.“<br>”. But, the program would produce integer 1 until 11 with twelve iteration if the \$i++ were before the echo \$i. “<br>”; in within the “statement execution”. This demonstrated that increment is an optional as an argument in within the for() loop argument. In addition, an infinity displayed of output would occur if there were no increment of value in the loop. Below show the details of the looping sequence in a flow chart form.

➤ The developed loop model:

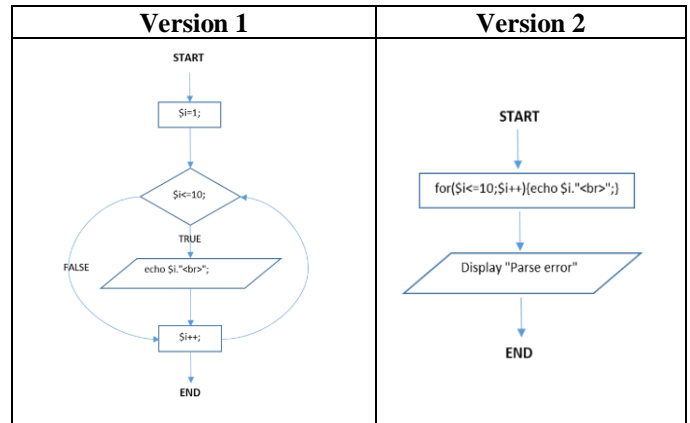


Fig 13:- The flow chart model of looping version 1 and version 2.

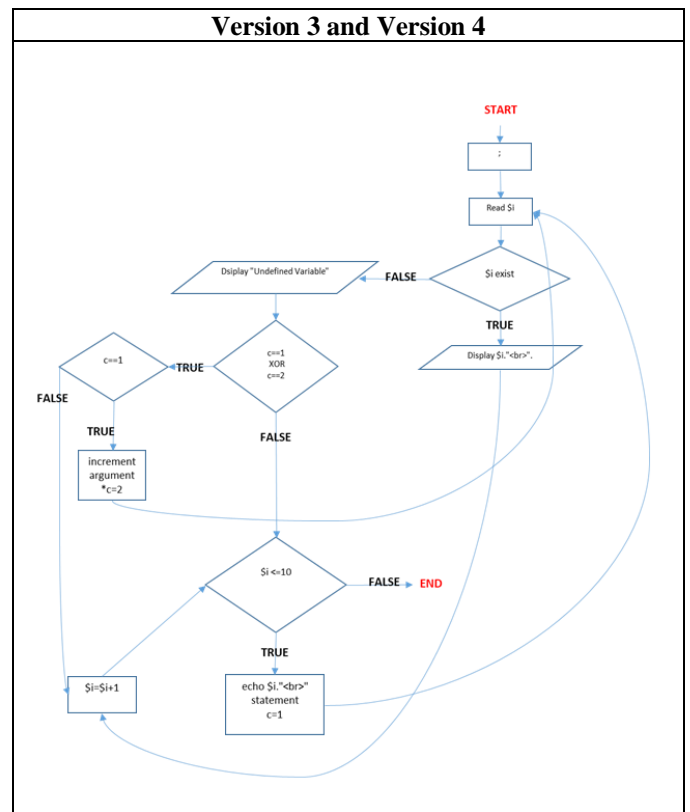


Fig 14:- The flow chart model of looping version 3 and version 4.

➤ *Graphical User Interface*

Below figures shows the output of all the versions of the program.

```
1
2
3
4
5
6
7
8
9
10
```

Fig 15:- Output of version 1.

```
Parse error: syntax error, unexpected ')', expecting ';' in C:\xampp\htdocs\loop.php on line 1
```

Fig 16:- Output of version 2.

```
Notice: Undefined variable: i in C:\xampp\htdocs\loop.php on line 3
Notice: Undefined variable: i in C:\xampp\htdocs\loop.php on line 3
Notice: Undefined variable: i in C:\xampp\htdocs\loop.php on line 3
1
2
3
4
5
6
7
8
9
10
```

Fig 17:- Output of version 3.

```
Notice: Undefined variable: i in C:\xampp\htdocs\loop.php on line 3
Notice: Undefined variable: i in C:\xampp\htdocs\loop.php on line 3
Notice: Undefined variable: i in C:\xampp\htdocs\loop.php on line 3
1
2
3
4
5
6
7
8
9
10
```

Fig 18:- Output of version 4.

➤ *Testing (Validation)*

Validation and Verification (V&V) were required in this research experiment in order to determine the output correctness and accuracy of the program. Therefore, the test case of the requirement were captures as a components and tested according to required test level and types which led to acceptance testing. Figure 19 to figure 38 shows the conducted testing activities in form of model.

➤ *Unit Testing*

```
1=x=<?php for($i=1;$i<=10;$i++){echo $i."<br>";} ?>
```

Fig 19:- Component version 1.

```
component(1=x=<?php for($i=1;$i<=10;$i++){echo $i."<br>";} )
→ result(y=1=figure15)
```

Fig 20:- Tested component of version 1.

```
1=x=<?php for($i<=10;$i++){echo $i."<br>";} ?>
```

Fig 21:- Component version 2.

```
component(1=x=<?php for($i<=10;$i++){echo $i."<br>";} ?>)
→ result(y=1=figure16)
```

Fig 22:- Tested component of version 2.

```
1=x=<?php for(;$i<=10;$i++){echo $i."<br>";} ?>
```

Fig 23:- Component version 3.

```
component(1=x=<?php for(;$i<=10;$i++){echo $i."<br>";} ?>)
→ result(y=1=figure17)
```

Fig 24:- Tested component of version 3.

```
1=x=<?php for(;$i<=10;){echo $i."<br>"; $i++;} ?>
```

Fig 25:- Component version 4.

```
→ result(y=1=figure18)
```

Fig 26:- Tested component of version 4.

➤ *Integration and System Testing*

**Version1:**

```
1=x=<?php for($i=1;$i<=10;$i++){echo $i."<br>";} ?>
```

Fig 27:- Component of version 1.

```
C1=component(1=x=<?php for($i=1;$i<=10;$i++){echo $i."<br>";} ?>)executed → result(y=1=figure15)
Figure 28. Integration Testing model with component executed.
```

$$\frac{C1}{result(y = 1 = figure15)}$$

Fig 29:- System Testing model with component as a system executed.

**Version2:**

```
1=x=<?php for($i<=10;$i++){echo $i."<br>";} ?>
```

Fig 30:- Component of version 2.

```
C1=component(1=x=<?php for($i<=10;$i++){echo $i."<br>";} ?>)executed → result(y=1= figure16)
```

Fig 31:- Integration Testing model with component executed.

$$\frac{C1}{result(y = 1 = \textit{figure16})}$$

Fig 32:- System Testing model with component as a system executed.

**Version3:**

```
1=x=<?php for(;$i<=10;$i++){echo $i."<br>";} ?>
```

Fig 33:- Component of version 3.

```
C1=component(1=x=<?php for(;$i<=10;$i++){echo $i."<br>";} ?>) executed → result(y=1= figure17)
```

Fig 34:- Integration Testing model with component executed.

$$\frac{C1}{result(y = 1 = \textit{figure17})}$$

Fig 35:- System Testing model with component as a system executed.

**Version4:**

```
1=x=<?php for(;$i<=10;){echo $i."<br>"; $i++;} ?>
```

Fig 36:- Component of version 4.

```
C1=component(1=x=<?php for(;$i<=10;){echo $i."<br>"; $i++;} ?>)executed → result(y=1= figure18)
```

Fig 37:- Integration Testing model with component executed.

$$\frac{C1}{result(y = 1 = \textit{figure18})}$$

Fig 38:- System Testing model with component as a system executed.

**IV. CONCLUSION AND RECOMMENDATION**

The importance of loop function in software based system is undeniable. This could be seen via a major system such as E-Commerce web system as well as small program which were used in education for teaching and learning purposes. In this research experiment, several different versions and designs of loop program were developed in order to understand the mechanism of the for() loop function in PHP, and it was found that the for() loop would require initialization and condition as their argument in order for the loop to be executed without any error. However, the initialization of value could be dismiss or absent in within the argument of for() loop, with a condition that “the end of line” (;) is written before the condition (\$i<=10;), as to tell the program that there is a line before the condition(\$i<=10;) and it could be the initialization, in addition, regards to the continuous execution even though without value assigned for initialization, the value of \$i in \$i<=10 could be treated as null or so called “zero” in terms of numerical in the first iteration, which allowed the for () loop function in this program to be executed until the final iteration. The incremental function would be an optional in within the argument which that its absentees does not lead to any error in the program execution. However, without increment, it would lead to infinity loop due to the condition (\$i<=10;) would be TRUE forever. In general, it is also suggested that more studies or research to be done towards the looping concept in order to understand deeply the loop function mechanism in variety of scripting or programming languages. This is important as knowing deeper or root about looping concept would lead to better understanding which could lead to better design [10].

**ACKNOWLEDGEMENT**

The main author would like to express his gratitude to the Office of Research Development & Consultancy of INTI International University (IU) and Faculty of Information Technology INTI IU for the supports in terms of resources allocation for the research as well as to Dr. A. Selamat (P.I.S), former associate researcher at the Institute for Mathematical Research (UPM) for important advice on research. The authors would like to express their gratitude to National Defence University of Malaysia for the priceless supports and motivation in doing research.

**REFERENCES**

- [1]. Halbert, D.C., 1984. Programming by example (Doctoral dissertation, University of California, Berkeley).
- [2]. Sommerville, I., 2011. Software engineering 9th Edition. ISBN-10, 137035152.
- [3]. Ahmad, D.K., Janicke, H., Shah, N.B.Z. and Onn, C.W., 2020. Perception of Software Agile Methodology Understanding among IT practitioner in Malaysia’s IT Industry. TEST Engineering & Management, 82, pp.12787-12795.

- [4]. Amadin, I.F. and Nwelih, E., 2010. An Empirical Comparison Of: HTML, PHP, COLDFUSION, PERL, ASP .NET, JAVASCRIPT, VBSCRIPT, PYTHON AND JSP. *Global Journal of Computer Science and Technology*, 10(12), pp.9-17..
- [5]. Civanlar, M.R. and Haskell, B.G., AT&T Corp, 1999. Client-server architecture using internet and public switched networks. U.S. Patent 5,995,606.
- [6]. Laurie, B. and Laurie, P., 2003. Apache: The definitive guide. " O'Reilly Media, Inc."
- [7]. Graham, D., Van Veenendaal, E. and Evans, I., 2008. Foundations of software testing: ISTQB certification. Cengage Learning EMEA.
- [8]. Lavalley, R. and Peacock, T.C., UNIVERSAL AUTOMATION Inc, 1989. Continuous flow chart, improved data format and debugging system for programming and operation of machines. U.S. Patent 4,852,047.
- [9]. Cantone, D., Omodeo, E. and Policriti, A., 2013. Set theory for computing: from decision procedures to declarative programming with sets. Springer Science & Business Media.
- [10]. Ciesielski, V. and Li, X., 2004, June. Experiments with explicit for-loops in genetic programming. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753) (Vol. 1, pp. 494-501)*. IEEE.