Asian Journal of Research in Computer Science

2(3): 1-5, 2018; Article no.AJRCOS.46136

A Comparative Analysis of Looping Structures: Comparison of 'While' Loop and 'Do-While' Loop in the C++ Language

P. Shouthiri[1*] and N. Thushika[1]

[1]Department of Mathematics, Eastern University, Sri Lanka.

Authors' contributions

This work was carried out in collaboration between both authors. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2018/v2i328752
Editor(s):
(1) Dr. Xiao-Guang Lyu, School of Science, Huaihai Institute of Technology, P.R. China.
(2) Dr. Peter Ayuba, Senior Lecturer, Faculty of Science, Department of Mathematical Sciences, Kaduna State University, Nigeria.
Reviewers:
(1) Iroju Olaronke, Adeyemi College of Education, Nigeria.
(2) Anthony (tony) Spiteri Staines, University of Malta, Malta.
(3) Oladele, Matthias Omotayo, The Federal Polytechnic, Nigeria.
(4) M. Bhanu Sridhar, GVP College of Engineering for Women, India.
(5) Pasupuleti Venkata Siva Kumar, Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology, India.
Complete Peer review History: http://www.sdiarticle3.com/review-history/46136

Original Research Article

Received 20 October 2018
Accepted 02 January 2019
Published 17 January 2019

ABSTRACT

Looping is one of the fundamental logical instructions used for repeating a block of statements. All programming languages are used looping structures to simplify the programs. Loops are supported by all modern programming languages, though their implementations and syntax may differ. This paper compares the two types of looping structures while and do-while using the compile time and runtime of the given programs to improve the efficiency of the programs. It is found that for small number of iterations while is efficient in runtime and do-while is efficient in compile time, the difference in total execution time may not be considerable. But given any large number of iterations, the difference is noticeable.

Keywords: Loop; while loop; do-while loop; compile time; run time.

**Corresponding author: E-mail: shouthiris@gmail.com;*

1. INTRODUCTION

A computer can process a program in one of the following ways in sequence, selectively, by making a choice as a branch, repetitively, by executing a statement again and again, using a structure called a loop; or by calling a function.

The most common control structures used in programming languages are selection and repetition. In selection, the program executes particular statements depending on different Boolean condition(s). In repetition, the program repeats particular statements a certain number of times based on different condition(s).

Loops are among the most elementary and powerful programming concepts in computer programming. In computer science, a loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Loops have been part of programming since the beginning of structured code in assembly language. In the earliest form using GOTO statements in assembly code, it is has now evolved to much simpler logical abstractions like 'while' and 'for' loops [1].

All Loops are supported and executed by all current programming languages, however it is obvious that, their syntax and implementation may differ. Three of the most common types of loops are the 'while' loop, 'do-while' loop and the 'for' loop [1].

Loops generally have three parameters:

- Initialize the loop control variable: Initialization should occur before the loop starts.
- Test the loop control variable: Testing is done within the parenthesis of the loop statement.
- Update the loop control variable: Updating is done somewhere inside the loop, usually as the last statement with in the loop.

The control variable initialization, though optional, is the condition that the loop must satisfy before it can begin with the first iteration of the loop. The variable update is optional in a looping structure [2].

The most significant parameter for a looping structure is the running condition. If the condition of running is correct or 'true' then the loop continues with the following iteration. Here, this condition is also optional, however, leaving this condition blank leads to the loop becoming an infinite loop. When the running condition statement returns incorrect or 'false', the next iteration is not begun and the loop statement is exited. This means that the code is allowed to proceed with the first statement occurring immediately after the loop statement.

The efficiency of a computer program highly depends on its run time and its compile time. Runtime is the duration of time when a program is running. It starts when a program is opened and ends with the program are quit. Compile time is the point at which a program is converted from source code to machine code. So, there is a need for comparing different looping structures in order to write efficient programs in computer science [3]. Presently, there are no enough comparisons made between 'while' and 'do-while' loops. This paper makes a comparison between the 'while' loop and 'do-while' loop to improve the efficiency of the programs.

2. WHILE LOOP

In most computer programming languages, a while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. A 'While' Loop is used to repeat a specific block of code an unknown number of times, until a condition is met.

The while construct consists of a block of code and a boolean condition.

```
while (boolean condition){
//statements
}
```

The boolean condition is evaluated, and if the boolean condition is true, the code within the block is executed. This repeats until the boolean condition becomes false. Because the while loop checks the boolean condition before the block is executed, the control structure is often also known as a pre-test loop [4].

3. DO-WHILE LOOP

A do-while loop is a control flow statement that executes a block of code at least once, and then repeatedly executes the block, or not, depending on a given boolean condition at the end of the block.

The do while construct consists of a block of code and a condition.

```
do {
// statements;
}while (boolean condition);
```

First, the code within the block is executed, and then the condition is evaluated. If the condition is true the code within the block is executed again. This repeats until the condition becomes false [4]. Because do while loops check the condition after the block is executed, the control structure is often also known as a post-test loop.

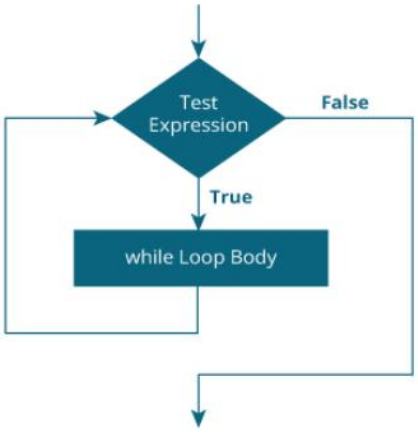

Fig. 1. Flowchart of while loop

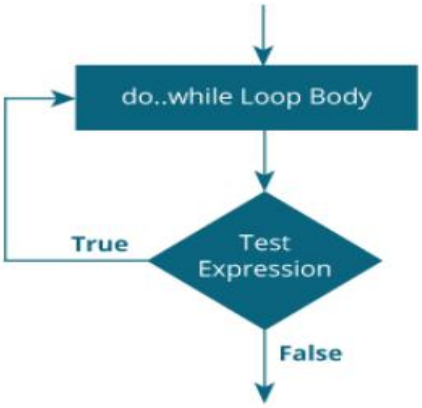

Fig. 2. Flowchart of do-while loop

4. COMPARISON OF WHILE AND DO-WHILE

Both while and do-while loops are the iteration statement, if we want it first, the condition should be verified, and then the statements inside the loop must execute then the while loop is used. If you want to test the termination condition at the end of the loop, then the do-while loop is used.

Significant Differences Between while and do-while Loop:

1. The while loop checks the condition at the opening of the loop and if the condition is fulfilled statement inside the loop, is executed. In do-while loop, the condition is checked after the execution of all statements in the body of the loop [5].
2. If the condition in a while loop is false, a single statement inside the loop is not executed [5], and if the condition in 'do-while' loop is false then also the body of the loop is executed at least once then only the condition is tested accordingly.

5. RESULTS AND FINDINGS

For comparing the execution time of while loop with the do-while loop, we checked the compile time and execution time of the programs which have written using while and do-while loop. Ten C++ programs were compared to analyze while and do-while loop.

1. Display even numbers from 1 – 100 (P1)
2. Display numbers from 100 to 0 in reverse order (P2)
3. Print the series 100, 95, 90, up to 5 (P3)
4. Find the given number is Prime or not (P4)
5. Find all square-numbers between 0 to 1000 (P5)
6. Display the reverse of given five-digit number (P6)
7. Display all numbers between 0 to 100 which are divisible by three and five (P7)
8. Display factorial of a given number (P8)
9. Find sum of odd numbers between two entered numbers (P9)
10. Display the series of Fibonacci for given number (P10)

Above programs written in DevCpp using while and do-while looping structures individually and the complilation time and running time were observed. Those values are tabulated in the Table 1.

The programs P1, P2, P3, to P10 which are written in C++ programming language are used to calculate the execution time and the compile time of the while and do-while loop. To calculate the compile time and execution time, DevCpp Integrated Development Environment (IDE) was used. According to the result compile time of the while is greater than do-while, but execution time of the while is less than do-while loop.

Table 1. Comparison of compile time and running time of the while and do-while loops

Program no.	Compilation Time (s)		Running Time (s)	
	While	do-while	While	do-while
P1	0.17	0.17	0.0488	0.0606
P2	0.19	0.19	0.1341	0.1593
P3	0.20	0.19	0.0235	0.0819
P4	0.21	0.19	2.0340	2.4620
P5	0.11	0.10	0.0073	0.0095
P6	0.13	0.13	0.0061	0.0167
P7	0.12	0.10	0.0065	0.0075
P8	0.18	0.17	1.4660	1.5790
P9	0.19	0.20	2.5580	2.7660
P10	0.19	0.18	1.5380	1.8650

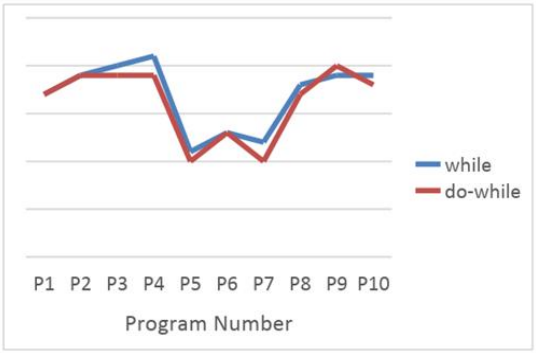

Fig. 3. Comparison of compile time of while and do-while loop

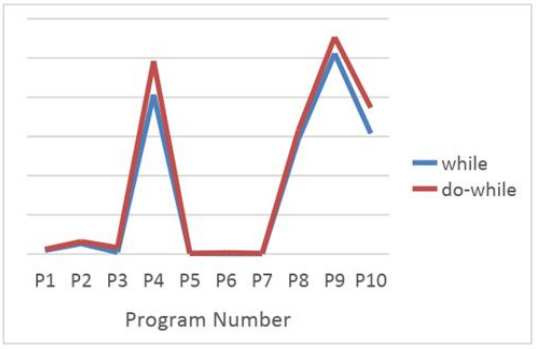

Fig. 4. Comparison of run time of while and do-while loop

Total execution time of the while and do-while loops were calculated by the sum of respective compile time and run time [6]. According to the comparison It is found that for small number of iterations while is efficient in runtime and do-while is efficient in compile time, the difference in total execution time may not be considerable. But for any given huge amount of iterations, the difference is fairly noticeable.

Table 2. Comparison of total execution time of the while and do-while loops

Program no.	Total execution time Time(s)	
	While	Do-while
P1	0.2188	0.2306
P2	0.3241	0.3493
P3	0.2235	0.2719
P4	2.2440	2.6520
P5	0.1173	0.1095
P6	0.1361	0.1467
P7	0.1265	0.1075
P8	1.6460	1.7490
P9	2.7480	2.9660
P10	1.7280	2.0450

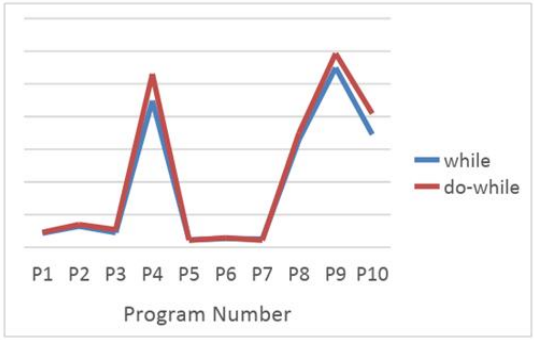

Fig. 5. Comparison of total execution time of while and do-while loop

6. CONCLUSION

These looping features enable programmers to develop concise programs containing repetitive processes that could otherwise require an excessive number of statements. It enables us to repeat a specific section of code or statement without the use of go to statements [7]. On some instances, it might be necessary to execute the body of the loop before the test is performed. Both while loop and do-while loops are the iteration statement, if we want that first, the condition should be clearly verified, and then the statements inside the loop must execute then the while loop is used. In order to test the termination condition at the end of the loop, then the do-while loop is used. Loops allow to repeat a block of code multiple times.

Both while and do-while are relatively similar in that both check a condition and execute the loop

body if it evaluated to true but they have one key difference. That is, a while loop's condition is checked before each iteration, the loop condition for do-while, however, it is checked at the end of each iteration. So, it is obvious that a do-while loop is always executed at least once. It is conceivable, and in some cases desirable, for the condition to always evaluate to true, creating an infinite loop. When such a loop is created intentionally, there is usually another control structure like break statement that allows termination of the loop.

7. FUTURE WORK

Looping control structures fully depends on 'for' loop, 'while' loop and 'do-while' loop. 'For' loop is more flexible to write as initialization, test condition, updating loop control variable all are written after 'for' keyword. So as a future work we are going to compare 'for' loop with 'While' and 'Do-While' loops with their processing time in order to increase a program's efficient to elicit effective program techniques.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Stroustrup B. Programming, 2nd ed. Upper Saddle River (New Jersey): Addison-Wesley; 2015.
2. Malik D. Introduction to C++ programming. 3rd Ed. Boston, MA: Course Technology, Cengage Learning; 2009.
3. En.wikipedia.org. Compile time; 2018. Available:https://en.wikipedia.org/wiki/Compile_ time [Accessed 10 Dec, 2018]
4. Asagba PO. Understanding C++ programming, Port Harcourt, Gitelle Press (Nig.) Ltd. 2002;157-175.
5. Hubbard JR. Programming with C++ Schaum's Outlines, New York, McGraw-Hill Companies, Inc. 2009;273–299.
6. Larson E. Program analysis too loopy? Set the loops aside. 11th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM). September 25-26, 2011;15-24.
7. Asagba, Prince Oghenekaro. A comparative analysis of structured and object-oriented programming methods. Journal of Applied Sciences and Environmental Management. 2008;12(4): 41-46.

Peer-review history:
The peer review history for this paper can be accessed here:
http://www.sdiarticle3.com/review-history/46136