SJIF Impact Factor 2021: 7.13| ISI I.F.Value:1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 6 | Issue: 2 | February 2021 - Peer Reviewed Journal

CASE STUDY ON THE EFFICIENCY OF AN EXECUTION OF NORMAL ARITHMETIC EXPRESSION AGAINST AN EXECUTION OF ARITHMETIC EXPRESSION IN LOOPING CONTROL STRUCTURE IN COMPUTER LANGUAGE

6243 – Cadet M Caleb Gunalan[1]
Class- XI 2020-21, Sainik School Amaravathinagar, Post: Amaravathinagar, Udumalpet Taluka, Tirupur Dt, Tamilnadu State

ABSTRACT

In computer science arithmetic operators are used to perform various arithmetic operations. The system performs particular operation depending on the type of the operator used in the expression.

All computer languages support all basic types of arithmetic operators, but the representation, meaning and order of execution of an operator in the expression solely depend on the basis of construction compiler or an interpreter.

This manuscript specifically examines the execution of direct method of arithmetic expression or normal arithmetic expression with an arithmetic expression using looping control structure in computer language, and to check how these two methods behaves, further comparing the efficiency of the these two approaches.

KEYWORDS: *Arithmetic Operators (AO), Unary Operators (UO), Binary Operators (BO), Runtime Execution(RE), O(n) Big O, Θ(n) Big Theta, Ω(n) Big Omega.*

1. INTRODUCTION

We have observed that in all the programming languages operators are used to perform operations on constant values and variables or in other words, operators are the tokens that perform some computation when applied to variable or constant values. The variables to which the computation is imposed are called the operands. The operators +, -,*, ÷ are most predominantly used arithmetic symbols in day to day life as well as in the computer system too. These operators are basic operators used to perform basic mathematical operations like addition, subtraction, multiplication and division respectively. There are several additional operators such as floor division operator, exponentiation operator, remainder operator.

2. CLASSIFICATION OF OPERATORS

Operators can be classified as Unary Operators(UO) and Binary Operators(BO). The unary operator are '+' and '-'. Always the unary '+' precedes an operand. When the two operands are involved in conjunction with the operator, classifies binary operator(BO). For example the two add two numbers we form an expression a+b, this + operator acts as binary operator in the given expression. The resultant of the expression will be the sum of two values which are represented in the form of operands.

This manuscript specifically examines the execution of direct method of arithmetic expression or normal arithmetic expression with an arithmetic expression using looping control structure, and to check how these two methods behaves, further comparing the efficiency of the these two approaches.

3. RELATED WORK

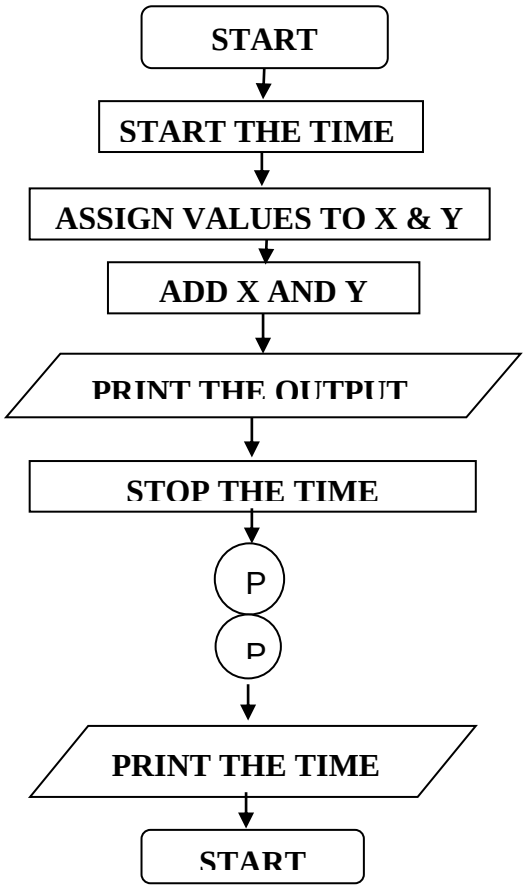
Algorithm for adding two numbers using normal arithmetic expression:

Step-1: Start
Step-2: import time module
Step-3: Start the time
Step-4: assign values to x & y

SJIF Impact Factor 2021: 7.13| ISI I.F.Value:1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 6 | Issue: 2 | February 2021 - Peer Reviewed Journal

Step-5: Add x and y
Step-6: Print the output
Step-7: Stop the time
Step-8: Print the time taken
Step-9: Stop

Flowchart for adding two numbers using normal arithmetic expression:



Snippet for adding two numbers using normal arithmetic expression:

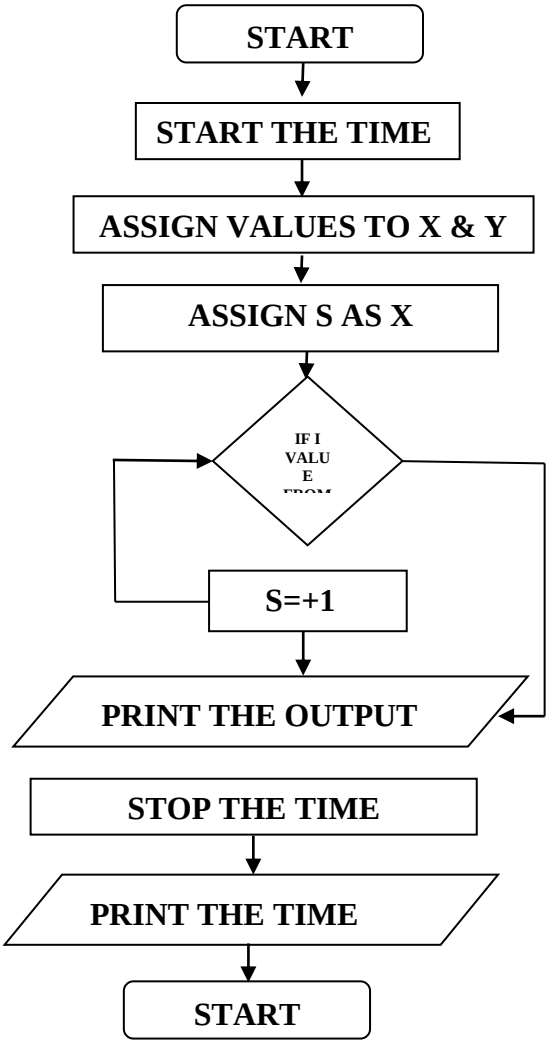
```
import time
start = time.time()
x=1
y=1
add=x+y
print("addition of two using + Operator number is :",add)
end = time.time()
print("Runtime of the program is:", end - start)
```

ARITHMETIC EXPRESSION IN LOOPING CONTROL STRUCTURE

Algorithm for adding two numbers using Looping Control Structure:

Step-1: Start
Step-2: import time module
Step-3: Start the time
Step-4: Assign values to x & y
Step-5: Assign s as x
Step-6: if i is in range 1 to y+1
Step-7: s will be increased by 1
Step-8: go to step 6(repeat if condition is true)
Step-9: Print the output
Step-10: Stop the time
Step-11: Print the time taken
Step-12: Stop

Flowchart for adding two numbers using Looping Control Structure:



Snippet for adding two numbers using Looping Control Structure:

```
import time
start = time.time()
x=1000
y=1000000
s=x
```

SJIF Impact Factor 2021: 7.13| ISI I.F.Value:1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 6 | Issue: 2 | February 2021 - Peer Reviewed Journal

```
for i in range(1,y+1):
    s+=1
print("addition of two number is using looping structure :",s)
end = time.time()
print("Runtime of the program is: ",end-start)
```

4. RUN TIME EXECUTION (RE) OF NORMAL ARITHMETIC EXPRESSION

USING NORMAL ARITHMETIC EXPRESSION		
X VAL	Y VAL	Time
0001	0001	0.03
2000	5000	0.034
1000	10000	0.049
1000	100000	0.054
1000	1000000	0.05

5. RUN TIME EXECUTION (RE) OF ARITHMETIC EXPRESSION IN LOOPING CONTROL STRUCTURE

USING LOOPING CONTROL STRUCTURE		
X Val	Y Val	Time
0001	0001	0.03
2000	5000	0.04
1000	10000	0.047
1000	100000	0.12
1000	1000000	0.61

6. SPACE COMPLEXITY AND TIME COMPLEXITY

In computer science, analysis of algorithms is a very crucial part. It is important to find the most efficient algorithm for solving a problem. It is possible to have many algorithms to solve a problem, but the challenge here is to choose the most efficient one.[1]

There are multiple ways to design an algorithm, or considering which one to implement in an application. When thinking through this, it's crucial to consider the algorithm's **time complexity** and **space complexity**.[2]

SPACE COMPLEXITY

The space complexity of an algorithm is the amount of space (or memory) taken by the algorithm to run as a function of its input length, n. Space complexity includes both auxiliary space and space used by the input.[2]

Auxiliary space is the temporary or extra space used by the algorithm while it is being executed. Space complexity of an algorithm is commonly expressed using Big (O(n)) notation.[2]

The Space complexity is ignored in this research paper, since the space complexity of particular problem is not considered so important.

TIME COMPLEXITY

The time complexity of an algorithm is the amount of time taken by the algorithm to complete its process as a function of its input length, n. The time complexity of an algorithm is commonly expressed using asymptotic notations:[2]

Big O - O(n)
Big Theta - Θ(n)
Big Omega - Ω(n)

It's valuable for a programmer to learn how to compare performances of different algorithms and choose the best time-space complexity to solve a particular problem in the most efficient way possible.[2]

Time Complexity for adding two numbers using normal arithmetic expression:

Big O notation is used in Computer Science to portrait the performance or complexity of an algorithm.

Big O specifically defines the worst-case scenario of an algorithm, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm. here O stands for order of growth.

Time Complexity for adding two numbers in using normal arithmetic expression **(worst case scenario)** is calculated as:

O(3)

Whereas, Time Complexity for adding two numbers using normal arithmetic expression **(worst case scenario)** is calculated as:

O(n)

CONCLUSION

The performance of these two methodologies exhibits that, the efficiency for calculating normal arithmetic expression is slightly higher when it is compared with the calculation time of arithmetic expression in the looping control structure. Further the worst case analysis is Big O (3) and Big O(n). In addition to this it is observed that the execution of expression also depends on the hardware configuration.

ACKNOWLEDGEMENT

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I express deep sense of gratitude to almighty God for giving me strength for the successful completion of the project.

SJIF Impact Factor 2021: 7.13| ISI I.F.Value:1.241| Journal DOI: 10.36713/epra2016 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Volume: 6 | Issue: 2 | February 2021 - Peer Reviewed Journal

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this project.

I express my deep sense of gratitude to the luminary **The Principal Capt (IN) Nirmal Raghu, Sainik School Amaravathinagar** who has been continuously motivating and extending their helping hand to us.

I express my sincere thanks to the academician The **Vice Principal Lt Col Nripendra Singh, Sainik School Amaravathinagar**, for constant encouragement and the guidance provided during this project.

I am overwhelmed to express my thanks to **The Administrative Officer Lt Col Amit, Sainik School Amaravathinagar** for providing me an infrastructure and moral support while carrying out this project in the school.

My sincere thanks to **Mr. Praveen M Jigajinni**, Master In-charge, A guide, Mentor all the above a friend, who critically reviewed my project and helped in solving each and every problem, occurred during implementation of the project

REFERENCES

1. *https://www.freecodecamp.org/news/time-complexity-of-algorithms/*
2. *https://www.educative.io/edpresso/time-complexity-vs-space-complexity.*