Otobo, D.W *et al*, International Journal of Computer Science and Mobile Applications,
Vol.10 Issue. 8, August- 2022, pg. 18-41 **ISSN: 2321-8363**
Impact Factor: 6.308
(An Open Accessible, Fully Refereed and Peer Reviewed Journal)

A Comparative Study of Parallel Programming Language

Otobo, D.W[1]; Anireh, V.I.E[2]
[1,2]Department of Computer Science, Rivers State University Port Harcourt, Nigeria
otobodina@yahoo.com
anireh.ike@ust.edu.ng
+234-7030367868, +234-8033229172

DOI: 10.5281/zenodo.6984851

ABSTRACT
This paper presents the Parallel Programming Languages and its Comparison of C and C++ programming Languages in terms of features, processing and development of high performance and efficiency of their running time. The C and C++ are programming Languages use for developing applications, games, database systems, operating systems and others. Both C and C++ seem similar but their features and usage are different. C is a procedural programming language that does not support objects and classes while C++ is an enhanced version of C programming with object-oriented programming support. But the significant difference between C and C++ is an object-oriented language that gives the advantages of data security, scalability, better performance and built-in functions.

Keywords: Parallelism, Programming, Algorithm

1.0 INTRODUCTION

Parallel programming languages are languages designed to program algorithms and applications on parallel computers. Parallel processing is of great value for developing high performance systems and solving large problems in many application areas. In recent years, parallel computers ranging from tens to thousands of computing elements became commercially available and continue to gain recognition as powerful tools in scientific research, information management and engineering applications. The trend of parallel programming languages and tools has contributed greatly to the usefulness of parallel computers specially in the areas of applications. Many models are being designed and implemented for the design and development of applications on parallel computers. Parallel programming languages are also known as concurrent languages. It allows the design of parallel algorithms as a set of concurrent actions that mapped into different computing elements. The cooperation between two or more actions can be performed in many ways according to the selected language.

2.0 LITERATURE REVIEW

Comparative Study of Parallel Programming Languages are important to determine the languages that can be of great benefits to the programmers or developers. It assists the programmer to be able to tell when a language is better as well as proffers valuable information during application and implementation. The comparatives between new languages and the predecessors are of vital roles in areas like speed, maintainability, functionality and performance of the languages. The parallel programming languages tools and it emergence has contributed immensely to the usefulness of parallel computers especially in the areas of applications. Different models are being designed and implemented for the design and development of applications on parallel computers. The distinguishing factors between parallelism and concurrency and how they overlap is the overview of existing parallelism technologies in C++ and the future directions being considered for parallel programming in standard C++.

C++ programming Language was developed by Bjarne Stroustrup a Danish Computer Scientist at bell Labs since 1979, for efficiency and flexibility and to provide high-level features for program organization. C++ is an extension of C, it was standardized by the International Organization for Standardization (ISO) (2020), with the latest standard version ratified and published by ISO as ISO/IEC 14882:2020 (informally known as C++20). The C++ programming language was standardized in 1998 as ISO/IEC 14882:1998, which was then amended by the C++03, C++11, C++14, and C++17 standards. The current C++20 standard supersedes the new features and an enlarged standard library. Bjarne Stroustrup (2013).

The evolution of C++ programming language, experiences some notable critics. Linus

Torvalds (88), Richard Stallman (89), Joshua Bloch, (90), Ken Thompson (91,92) and the Donath Knuth (93, 94). The critics were based on the complexity of the language and the large number of non-orthogonal features which was a restriction to coding. But was rectified at the ISO working group to help programmers write modern C++ by using C++14, until the current C++20, Joshua Bloch (2001).

Researchers have paid much attention to parallel programming Language since the emergence of Pablo Halpern (1989) programming in C++, a member of the C++ Standards Committee since 2007. He is currently the Parallel Programming Languages Architect at Intel Corp., where he coordinates the efforts of teams working on Cilk Plus, TBB, OpenMP, and other parallelism languages, frameworks, and tools targeted to C++, C, and Fortran users. His current work is focused on creating simpler and more powerful parallel programming languages and tools for Intel's customers and promoting adoption of parallel constructs into the C++ and C standards.

Parallel Programming Language Aids

1. It help to decrease High -Level Language, both the design time and the execution time of parallel applications, and make it easier for new users to approach parallel computers.
2. It aids the software in making quick decisions, through parallel programming in C and C++ and
3. Multithreading (multithreaded programming).
4. It speed-up the program from parallelization and limit the program to a parallelize state.

Features of C

Simple	C programming is one of the oldest programming languages and its simple and easy -to -understand.
Procedural Language	The C programming uses special flow of the program to run the code. It breaks the code into small blocks for different functions to minimizes the complexities.
Case Sensitive	The C programming Language is case sensitive.

Portable	The C programming Language is portable or adaptable to different platforms and systems.
Dynamic Memory Allocation	C support dynamic memory allocation. ie allowing memory space during the run time as well.
Speed	C programming Language is faster, compares to other languages like Java and Python.
Rich Library	C programming Language gives a library with built-in functions or users -defined functions.

Features of C++

Fast and Powerful	C++ programming Language is a compiler -based Language. It enhances and makes the execution of the code faster.
Pointers	The C++ programming Language uses pointers to hold the addresses of an object. This features allows the programmers to use pointers for interacting with the functions, arrays, memory and structures with less code.
Static Type System	The C++ programming Language is a compiler-based Language.
Object-Oriented Language	The C++ programming Language is an extension to the procedural programming language C, with the object - oriented programming concepts, which includes Polymorphism, Encapsulation, Abstraction, inheritance etc. These features help in code maintenance, prevention of data redundancy and ensure flexibility and effective problem solving.
Additional features	C ++ is a superset of the programming language C, which includes all the features of C as well e.g dynamic allocation, rich library portability and structured programming. etc.

Difference between C and C++

Parameter	**C**	**C++**
programming style	The C programming language is a procedural language types, and follows a top to down approach.	C++is an object –oriented programming language type and is bottom –to-top approach.
Approach	Programming approach that focuses on the steps rather the data.	focuses on the data rather than the overall procedure.
Program division	C is a structured programming language the program is divided into separate blocks known as functions.	C++ is an object-oriented programming language, the code is divided into object and classes.
Data types	C is a basic version of programming language that supports only primitive, fixed data types.	C++is an enhance version of C and supports generic data types.
Exception Handing	C does not supports exception handing, i.e., supports in times of 'hard' errors causing code problems.	C++ supports exception handing and provides efficient support during errors an incorrect code.
Application Development	This programming language is more suitable for assemblers, text editors, network driver, and low-level implementations.	C++ programming language is suitable and extensible for high-end programming including game development, embedded systems like smart watches, medical machines etc.

Compatibility	C is the fundamental language and the code written with C can be run with the C++ compiler.	C++ is the superset of the C language including OOP concepts and hence, cannot run the code in the compiler.
File Extension	C is the file extension for the C programming language.	C++ is the file extension for C++.
Ease of Coding	C is known as Hands-on language which means C allows the programmer to tell everything and it is easy.	C++ is a more object-Oriented high-level programming language that requires fixed construction and principles and it is easier to code.
Data Security	C programming language does not adhere to the encapsulation concept and allows easy data manipulation from outside code.	C++ is a more secure programming language.
Inline Function	C does not support Inline Function	C++ supports Inline Function
Variable	A variable is like a storage location in C, it needs to be defined at the beginning itself.	Variable in C++ can be declared anytime.
Namespace	To organize the code for efficiency and prevent collisions, a namespace is required. C does not support that.	C++ as a flexible programming language supports namespace.
Source code	C is the base of many foundational languages and is itself known for its free format source code.	C++ was developed, inspired by the C++ programming language.

Used by	Microsoft Windows Kernel, Telegram Messenger, Oracle Database, MySQL, etc.	Google Chrome, Microsoft Office, Torque 3-D game engine, and so many more.

1. Id: A language with implicit parallelism

The Id as a parallel programming language developed in the Computation Structures Group at MITs Laboratory for Computer Science. There are three major goals in developing Id that is

(1) High level,
(2) General purpose, and
(3) High performance.

High level is expressive as the modern functional languages and Lisp. Parallelism in Id is implicit, the user does not have to manage partitioning, scheduling, and synchronization. General purpose is suitable for both scientific and symbolic computation.

Id has efficient arrays and floating-point operations, as well as recursive data structures in an automatically managed heap. High performance for an Id program compiled for a dataflow machine to achieve at least as much absolute performance as its FORTRAN counterpart on a Von Neumann machine built with comparable technology. Id is a layered language, with layer 0 representing the cleanest semantics and layer 2 representing the most expressive power. Layer 0 is purely functional, and is similar to other modern functional languages like Miranda and Haskell, it has higher-order functions, non-strict semantics, polymorphic types with static type-checking by inference, algebraic types with pattern-matching, list comprehensions, and user-defined abstract data types. Id's array comprehensions are fairly unique.

S/n	**Description**	**Id**
1	High Level	Is expensive as the model functional Language.
2	General Purpose	Suitable for both Scientific and symbolic computation
3	High Performance	Id program through data flow machine
4	Parallelism	Is implicit, user does not have to manage partitioning, scheduling and synchronization.

Parallelism

Parallelism is the process of processing several set of instructions simultaneously.

1. It reduces the total computational time.
2. Parallelism can be implemented by using parallel computers i.e. a computer with many processors.
3. Parallel computers requires parallel algorithm, programming Languages, compilers and operating system that support multitasking.

Features for Parallel programming Language

Languages for parallel programming should meet four goals:

- expressiveness
- reliability
- security and
- verifiability.

The C* Parallel Programming Language

C* assumes the same abstract machine model as C sharing the features of

(1) a uniform address space

(2) sequential execution, and

(3) pointer arithmetic.

The only difference in the models arises from C*'s execution of the instruction stream by many processors instead one. This difference is a key feature of the C* language which seeks to assign a processor to every data item. Just as a well written C program scales with the size of arrays, C* scales by assuming a virtual processor is assigned to every point in an array. When more virtual processors are required than the actual number of physical processors requested, multiple virtual processors are assigned to each physical processor. The number of virtual processors that can be assigned to a physical processor is limited only by local memory.

Classification of Parallel Programming Language

There are 4 types of Programming Language

- Procedural Programming Language.
- Functional Programming Language.
- Scripting Programming Language.
- Logic Programming Language.
- Object-Oriented Programming Language.

Procedural Programming	**Functional Programming**	**Scripting Programming**	**Logic Programming**	**Object Oriented Programming**
C++	**Id**	**PHP**	**Prolog**	**Ada**
SQL	**Haskell**	**JavaScript**	**Answer set programming (ASP)**	**C***
BASIC	**Closure**	**Python**		**C++**
FORTRAN	**Concurrent ML**	**jQuery**	**Datalog**	**JS**
JAVA	**Multi Lisp**			**C#**
PASCAL				**Java**
				Python

Parallel programming models:

Name	**Class of interaction**	**Example implementations**
Functional	Message passing	Concurrent Haskell, Concurrent ML
LogP machine	Synchronous message passing	None
Parallel random access machine	Shared memory	Cilk, CUDA, OpenMP, Threading Building Blocks, XMTC

The best language for parallel programming

The best parallel Programming languages, are, C and C++, they have evolved to make it easier to use multiple threads and handle complexity. Both C and C++ now includes threading libraries. Modern C++, in particular has make parallel programming easier. The inclusion of a standard threading library C++11.

Parallel processing

In a typical computer setting, a complex task are divided into multiple parts with a software tool and are assign to each part to a processor to solve its part, and the data is reassembled by a software tool to read the solution or execute the task.

The difference between Parallel Programming and Multithreaded Programming

Parallel programming is a broad concept. It describes many types of processes running on the same machine or on different machines. While, multithreading specifically, refers to the concurrent execution of more than one sequential set (thread) of instructions.

Multithreading programming is programming multiple, concurrent execution threads. These threads could run on a single processor or there could be multiple threads running on multiple processors cores.

- **Multithreaded Programming on a Single Processor**

Multithreading on a single processor gives the illusion of running in parallel. In reality, the processor is switching by using a scheduling algorithm. Or, it's switching based on a combination of external inputs (interrupts) and how the threads have been prioritized.

- **Multithreaded Programming on Multiple Processors**

Multithreading on multiple processor cores is really parallel. Individual microprocessors can work together to achieve the result more efficiently. And also multiple parallel, concurrent tasks can happen once.

Importance of Multithreaded Programming

Multithreading is important to development teams today as technology evolves.

- **Processors Are at Maximum Clock Speed**

Processors have reached maximum clock speed. The only way to get more out of CPUs is with parallelism.

Multithreading allows a single processor to spawn multiple, concurrent threads. Each thread runs its own sequence of instructions. They all access the same shared memory space and communicate with each other. The threads can be carefully managed to optimize performance.

Importance of Multithreading Libraries to C and C++ language

Moving from single-threaded programs to multithreaded increases complexity. Programming languages, such as C and C++, have evolved to make it easier to use multiple threads and handle this complexity. Both C and C++ now include threading libraries.

Modern C++, in particular, has make parallel programming easier. The inclusion of standard threading library C++11. In C++17 has added parallel algorithms and parallel implementations of many standard algorithms. An advantages for parallelism is in future versions of C++.

3.0 Efficiency and speed-up in C# and C++ programming Language

- One of the reasons C++ is faster. C++ compilers have been for past 35 years. C# compilers has get better over time.

C++ code used to be significantly faster for a long time, and also today still is in many cases. This is mainly due to the more advanced JIT optimizations being complicated to implement.

C++ is faster, where C++ is actually faster, are highly optimized programs, where expert programmers thoroughly optimized the hell out of the code. This is not only very time consuming (and thus expensive), but also commonly leads to errors due to over-optimizations.

Code in interpreted languages gets faster in later versions of the runtime (.NET CLR or Java VM) due to the usefulness of optimizations JIT compilers .These are simply impossible in languages with pointers. Also, some argue that garbage collection should generally be as fast or faster as manual memory management. Generally implementation can be achieve in C++ or C, but much more complicated and prone to error.

If the application will mostly consist of very performance critical arithmetic, and that it will be the bottleneck, and it is certainly going to be faster in C++, finding the performance bottlenecks if it runs too slow, then think about how to optimize the code. In the worst case, you might need to call out to C code through a foreign function interface, you can still have the ability to write critical parts in lower level language.

C++ that really depends on the libraries been in use, C# is not typically faster, but .NET which is applicable using Python instead of C to write some code.

C and C++ main problem was pointer management which is pretty much resolved with smart pointers. C++ lacks an extensive library as .NET and Java offers and that can speed up development largely.

3.1 Faster implementation can be created in C++ than in C#.

There are so many variables like the task, problem domain, hardware, quality of implementations, and many other factors thereby it cannot really be ascertained.

C#'s runtime optimizer are very good, and is able to perform certain dynamic optimizations at runtime that are simply not available to C++ with its compile-time (static) optimizer.

3.2 C++ and its performance.

C++ are still very good when polymorphic decisions can be predetermined at compile time.

Generally, encapsulation and decision-making is a good thing because it makes the code more dynamic, easier to adapt to changing requirements and easier to use as a framework. Object oriented programming in C# are very productive and it can be generalized under the term “generalization”.

C/C++ can perform vastly better in programs where there are either large arrays or heavy looping/iteration over arrays (of any size). In graphics they are much faster in C/C++, because heavy array operations underlie almost all graphics operations. .NET is slow in array indexing operations due to all the safety checks, and is especially true for multi-dimensional arrays such as rectangular C# arrays.

C, C++, Java, and C# to implement the exact same algorithmic program in the latter 3 languages. The program had a lot of math and multi-dimensional array operations. Java was about 1.3x faster than C# (most JVMs are more optimized than the CLR), and the C++ raw pointer version came in about 2.1x faster than C#.

.NET languages can be as fast as C++ code, or even faster, but C++ code will have a more constant throughput as the .NET runtime has to pause for GC,

Applications that require intensive memory access e.g. image manipulation are usually better off written in unmanaged environment (C++) than managed (C#). Optimized inner loops with pointer arithmetic are much easier to have control of in C++.

3.4 Comparisons in algorithmic complexity between the languages

The time complexity for a given program

Time complexity represents the number of time a statement is executed. The time complexity of an algorithm is not the actual time required to execute a particular code, since that depends on other factors like programming language, operating software, processing power etc. The idea behind time complexity is that, it can measure only the execution time of the algorithm in away that depends only on the algorithm itself and its input.

1. **How do we calculate Time Complexity**
   The time complexity, measured in the number of comparisons, then becomes T(n) = n-1. General, an elementary operation must have two properties. There can not be any other operations that are performed more frequently as the size of the input grows.

2. **The complexity of a program in C**

   Time complexity of program is calculated based on the number of input for the program. Given, the complexity is always constant so the time complexity is O(1).

The time Complexity of a given program C

```
# include <stdio.h>

#include < conio.h>

Void main ()   {

  Int  a,b, sum,

Clrscr();

Print (“Enter  2  numbers  “);

Scanf (‘%d%d”, & a& b);

Sum = a+ b;

Printf (“sum of 2 numbers;%d”, sum);

Getch ();

}
```

The time complexity of a C++ program

The inner loop is executing (log n) times where the outer is executing n times. For a single value of i, j is executing (log n) times, for n values of i, j will loop total n*(log n) = (n log n) times. Therefore, the time complexity is O(n log n).

1. **The time complexity C++:**
Time complexity is the amount of time taken by an algorithm to run, as a function of the length of the input. It measures the time taken to execute each statement of code in an algorithm.

2. Time Complexity Analysis in C++

Time complexity of any algorithm is the time taken by the algorithm to complete. It is an important metric to show the efficiency of the algorithm and for comparative analysis. Reducing the time complexity of algorithm makes it more effective.

Example 1

Find the time complexity of the following code snippets

```
for(i= 0 ; i < n; i++){

  cout<< i << " " ;

  i++;

}
```

The loop has maximum value n but the i will be incremented twice in the for loop which will make the time take half. Therefore, the time complexity is O(n/2) which is equivalent to O(n).

Example 2

Find the time complexity of the following code snippets

```
for(i= 0 ; i < n; i++){

for(j = 0; j<n ;j++){

 cout<< i << " ";

 }

}
```

The inner loop and the outer loop both are executing n times. So for single value of i, j is looping n times, for n values of i, j will loop total n*n = n 2 times. So the time complexity is O(n 2).

Example 3

Find the time complexity of the following code snippets

```
int i = n;

while(i){

cout << i << " ";

  i = i/2;

}
```

In this, after each iteration the value of i is turned into half of its previous value. Therefore, the time complexity is O(log n).

Example 4

Find the time complexity of the following code snippets

```
if(i > j ){

  j>23 ? cout<<j : cout<<i;

}
```

There are two conditional statements in the code. Each conditional statement has time complexity = O(1), for two of them it is O(2) which is equivalent to O(1) which is **constant**.

Example 5

Find the time complexity of the following code snippets

```
for(i= 0; i < n; i++){
 for(j = 1; j < n; j = j*2){
  cout << i << " ";
}
}
```

The inner loop is executing (log n) times where the outer is executing n times. Therefore, single value of i, j is executing (log n) times, for n values of i, j will loop total n*(log n) = (n log n) times. So the time complexity is O(n log n).

4.0 Performance is slow compared to C++.

C++ language is an object-oriented programming language, it supports some important features like Polymorphism, Abstract Data Types, Encapsulation, etc. Since it supports object-oriented, speed is faster compared to the C language.

C and C++ programming Basics

Identical Fibonacci loop programs for both c and c++ to test relative speed. The c++ version is 60x slower. All they do is loop through and print the first 14 Fibonacci numbers 10,000 times.

The c version:

```
   #include <stdio.h>

   int main (){

     int c = 0;

    int x, y, z;

    while(c < 10000)

   {

     x = 0;

    y = 1;

   while(x < 255)

    {

printf("%d\n", x);

z = x + y;

x = y;

y = z;

 }

 c++;

}
```

```
    return 0;
}
```

and here is the c++ version:

```
#include <iostream>
using namespace std;
int main()
{
    int c = 0, x = 0, y = 0, z = 0;
    while(c < 10000)
    {
        x = 0;
        y = 1;
        while(x < 255)
        {
            cout << x << endl;
            z = x + y;
            x = y;
            y = z;
```

```
      }
      c++;
    }
    return 0;
}
```

4.0 C was developed by Dennis Ritchie between the year 1969 and 1973 at AT&T Bell Labs.	C++ was developed by Bjarne Stroustrup in 1979.
C does no support polymorphism, encapsulation, and inheritance which means that C does not support object oriented programming.	C++ supports polymorphism, encapsulation, and inheritance because it is an object oriented programming language.
C is a subset of C++.	C++ is a superset of C.
C contains 32 keywords.	C++ contains 63 keywords.
For the development of code, C supports procedural programming.	C++ is known as hybrid language because C++ supports both procedural and object oriented programming paradigms.

Data and functions are separated in C because it is a procedural programming language.	Data and functions are encapsulated together in form of an object in C++.
C does not support information hiding.	Data is hidden by the Encapsulation to ensure that data structures and operators are used as intended.
Built-in data types is supported in C.	Built-in & user-defined data types is supported in C++.
C is a function driven language because C is a procedural programming language.	C++ is an object driven language because it is an object oriented programming.
Function and operator overloading is not supported in C.	Function and operator overloading is supported by C++.
C is a function-driven language.	C++ is an object-driven language
Functions in C are not defined inside structures.	Functions can be used inside a structure in C++.
Namespace features are not present inside the C.	Namespace is used by C++, which avoid name collisions.
Header file used by C is stdio.h.	Header file used by C++ is iostream.h.
Reference variables are not supported by C.	Reference variables are supported by C++.

Virtual and friend functions are not supported by C.	Virtual and friend functions are supported by C++.
C does not support inheritance.	C++ supports inheritance.
Instead of focusing on data, C focuses on method or process.	C++ focuses on data instead of focusing on method or procedure.
Cprovides malloc() and calloc() functions for dynamic memory allocation, and free() for memory de-allocation.	C++ provides new operator for memory allocation and delete operator for memory de-allocation.
Direct support for exception handling is not supported by C.	Exception handling is supported by C++.
scanf() and printf() functions are used for input/output in C.	cin and cout are used for input/output in C++.
C structures do not have access modifiers.	C ++ structures have access modifiers.

5.0 Conclusion:

- Both the languages have a similar syntax.
- Code structure of both the languages are same.
- The compilation of both the languages is similar.
- They share the same basic syntax. All of C's operators and keywords are also present in C++ .
- C++ has a slightly extended grammar than C, but the basic grammar is the same.
- Basic memory model of both is very close to the hardware.
- Same notions of stack, heap, file-scope and static variables are present in both the languages.

Differences between C and C++

C++ can be said a superset of C. The major added features in C++ are Object-Oriented Programming, Exception Handling and rich C++ Library.

REFERENCES

[1] Becker, Pete (2006). *The C++ Standard Library Extensions : A Tutorial and Reference*. Addison-Wesley. ISBN 0-321-41299-0
[2] Bridonce P. (2021). Parallel Programming, *Retrieval from http://www.science.direct.com.*
[3] *Bjarne Stroustrup. "A history of C++: 1979-1991". doi:10.1145/234286.1057836.*
[4] *Bjarne Stroustrup's Homepage". www.stroustrup.com.*
[5] Chandra, R., Dagum, L., Kohr, D. (2000): Parallel Programming in OpenMP++. Morgan Kaufmann, San Francisco
[6] Chandy, K. M., Taylor, S.(1991).*An Introduction to Parallel Programming, Jones and Bartlett, 1991.*
[7] *Dewhurst, Stephen C. (2005). C++ Common Knowledge: Essential Intermediate Programming. Addison-Wesley. ISBN 0-321-32192-8.*
[8] *Information Technology Industry Council (2003). Programming languages C++ (Second ed.). Geneva: ISO/IEC. 14882:2003(E).*
[9] John .T. FEO (1992). *A comparative study of Parallel Programming Languages (the Salishan problems)*
[10] Jones, S.P. (ed.): Haskell 98 language and libraries. Journal of Functional Programming 13(1), 1–255 (2003)
[11] *Josuttis, Nicolai M. (2012). The C++ Standard Library, A Tutorial and Reference (Second ed.).Addison-Wesley. ISBN 978-0-321-62321-8.*
[12] *Mycroft, Alan (2013). "C and C++ Exceptions Templates" (PDF). Cambridge Computer Laboratory - Course Materials 2013-14.*
[13] *Naugler, David (2007). "C# 2.0 for C++ and Java programmer: conference workshop". Journal of Computing Sciences in Colleges.* ***22*** *(5).*
[14] Parallel Algorithm- *Retrieval from, http://www.tutorials point.com*.
[15] *Stroustrup, Bjarne (2000). The C++ Programming Language (Special ed.). Addison-Wesley. p. 310. ISBN 0-201-70073-5.*
[16] *Stroustrup, Bjarne (2013). The C++ Programming Language. Addison Wesley. pp. 345, 363. ISBN 9780321563842*
[17] *Stroustrup, Bjarne (2013). The C++ Programming Language. Addison Wesley. pp. 363–365. ISBN 9780321563842.*
[18] Stroustrup, Bjarne (2014). *Programming: Principles and Practice Using C++* (Second ed.). Addison-Wesley. ISBN 978-0-321-99278-9
[19] Stroustrup, Bjarne (2010). "Bjarne Stroustrup's FAQ:
[20] Time Complexity Analysis, *Retrieval from. http://www.tutorialspoint.com*
[21] Understanding Time Complexity *with simple examples Retrieval from, https://www.geeks forgeeks.org.*