D- A Powerful General Purpose Application Programming Language

Tarun Kumar
Assistant Professor
Vidya College of Engineering
Bagphat Road, Meerut (India)

Mayank Singh
Associate Professor
Krishna Engineering College
Mohan Nagar, Ghaziabad (India)

Arun Sharma
Professor& Head, CSED
KIET, Ghaziabad (India)

ABSTRACT

The real purpose of a programming language is to communicate a set of instructions to a system, basically a computer. Just like natural languages, programming languages are described using their syntaxes and semantics. This paper presents a comparison of a newly-developed Programming Language called D with traditional languages like C, C++, C# and JAVA. D is applicable to many programming paradigms like object-oriented with classes and interfaces, generic programming with templates and mix-ins, procedural programming with functions and arrays etc. There are two major versions of this language - D1 and D2. This paper also presents a comparison of the execution times of a program, which is written in the C, JAVA, C# and D languages. This is a comparative study, which illustrates the behavior of these languages using a program structure in each of the languages. This is done by developing a sample program in which various languages are used to check for the difference in flow of execution. This will firstly provide an opportunity to examine the variation in programming styles of different programmers. Second, it permits accessing and comparing the variability of program properties induced by different languages. Results indicate that for the given programming problem, D is more productive compared to other languages like C, C++, C# and Java.

Keywords

C, C++, C#, Java, D, and Execution Time etc...

1. INTRODUCTION

BCPL stands for Basic Combined Programming Language. It was designed by Martin Richard who was associated with the University of Cambridge in 1966. It works with the approach of structural, procedural and imperative paradigm.

The fundamental planning behind the initiation of BCPL was for developing compilers for other languages, which is less frequently use now. The extended version of BCPL is called B, which was used for developing the UNIX operating system. Later on the B language become the core of the C language. The aim of a programming language is to synchronize the operations that are performed by a computer. There are thousands of new programming languages on which programmers are working to develop new software; but only a few of them popular to be utilized in the long run of development. Simply working with one programming language may not support all the requirement of the users. There must be a multi-functional programming platform which is support all the requirement of the users by using the different syntax. Generally there are two paradigms to develop any application specifically the function-oriented and object-oriented language. Both paradigms have their own limitations. The D language is a combination of a procedural and an object-oriented language.

D is an application-based language, which is used for general-purpose system programming, providing an interface between hardware and application programming. It provides features such as first-class array, inline assembler, lack of preprocessor, contracts, a garbage collector and many more. D is a multi-paradigm, natively-defined, compiled, and statically-typed language. The motive behind the D language is to develop any type of application by combining the syntaxes of flexible C and JAVA languages.

The first ANSI C++ compiler for DOS was written by Walter Bright who originally conceived the D language. Compliers of the D language are freely downloadable for Windows and Linux.

Over the last fifty years, programming languages have taken a great leap in terms of development as well as their utilization. In comparison to the advent of UNIX and C, when compiler translates these languages they were just getting their initial start, the latest features of the languages are fulfilling the requirements of different development strategies [2].

D is well fitted for writing small-scale to large-scale programs. With a group of programmers, it is very easy to learn and utilize many features of programs, which are suited for aggressive compiler optimization technology. This language fulfills different requirements in the programming community. D can be compared with Ruby; python the basis of SHI syntax, array and type inference and it is also available for low level system programmer with features such as inline assembler.

It brings in features of imperative languages, such as Lisp, with the lazy storage class, which drastically speeds up efficiency. The language is relatively stable, with the occasional new features or changes in it.

D is basically used by a programmer for coding high performance and reliable code, which wants a C++/JAVA style language, but requires a special language which is much easier to master with support of modern techniques and management like a D program as it contains comparatively less code. It is not unusual for a D program to contain 30% less source code as compared to an equivalent C++ source code, yet it runs at the same speed or faster. It is manageable to develop and debug code in D platform at a fast speed. The D language was designed with the following rules in mind:

1) Text pre-processor is not required.
2) Command line switch cannot change the syntax
3) Imported file cannot change the syntax
4) A source code can be tokenized without direction to semantics or syntax
5) Analysis of source file can be performed

In short, D is a language ready for real-world deployment.

2. GENERAL COMPARISON

Table-1 compares general and technical information for a selection of common- used programming languages [1] [4-7].

Table 1: A General Comparison

Language	**Intended Use**	**Paradigm(s)**
C	System	Imperative, Procedural
C++	Application, System	Generic, Imperative, Object-Oriented, Procedural, Functional
Java	Application, Business, Client-side, General, Server-side, Web	Reflective , Generic, Imperative, Object-Oriented,
Python	Application, General, Web, Scripting	Aspect-oriented, Functional, Imperative, Object-oriented, Reflective
PHP	Server-side, Web Application, Web	Reflective, Imperative, Object-Oriented, Procedural,
C#	Software components, hosted and embedded systems	multi-paradigm: structured, imperative, object-Oriented, event-driven, functional, generic, reflective, concurrent
D	Application, System	Generic, Generative, Imperative, Object-oriented, Functional, Concurrent

3. D LANGUAGE

GDC compiler utilizes the GCC with the back end, built using the open DMD compiler source code. D's key features retain C++'s ability to do low-level coding, support memory safe programming and multi-paradigm programming, has a garbage collector, flexible first-class arrays, implements design by contracts, integrated inline assembler, etc. The D language also exhibits application programming interface compatibility with C, thereby permitting usage of all the old C libraries and being shorter and memory-safe as well. All standard C, C++ and Java functions are a part of the D standard library [2].

(1) In D, the ‘Welcome in D’ program:

```
import std.stdio;  // standard I/O module
int main (char [] [] args)
{
    writefln (" First Step  in D!");
    return 0;
}
```

(i) printf id d is represented as writef.

(ii) writefln is used as a newline character at the end.

(2) Alias and typedef

Alias is providing an alternative name to an already existing typedef which considers only types and develops a new type [8]:

```
alias int size_r;
typedef int mynumeric;
alias someReallyLongFunctionName func;
```

(3) Arrays

Three types of arrays hold with the D language: associative arrays, static and dynamic array. Fashion of declarations of array is right to left [8].

```
char [] [] is understood as a collection of characters:
int [] array_namr; // dynamic array of ints
int [2] [4] array_name; // a 2x4 matrix
```

Sort, reverse and length properties also exit in array with the D language. The Operator slices with Dynamic and static array.

```
int [] num = [15, 22, 32, 42, 50, 62, 72];
num = num [0..20]
```

(4) ~ operator

Tidle ~ operator in the D language is used for concatenation and most important fundamental concept of concatenation is the string [3].

```
char [] string1 = "Welcome ";
char [] string2 = "In D!";
char [] string = string1 ~ string2; // Welcome in D!
```

Additionally, there are three types of string based on UTF-8, 16, and 32 bit systems on D platform.

```
char;
wchar, and
dchar;
```

D is a suitable language to internationalized programming due to different kind of supporting Unicode protocol, wherein the Unicode characters are manipulated using the standard and library routines.

(5) Unit Test

Unit testing is a main feature of D. The purpose of Unit Testing is to ensure that a particular function or set of functions is working according to specification with various possible arguments [3]:

```
int add (int a, int b)
{
  return a + b;
 }
```

The unit test would be placed in the same module, and if the unit test option is enabled, it would run as soon as the module is imported and any function from it is executed. In this case, it probably would look something like this:

```
Unit_test
{
      assert (add (1, 2) == 3);
      assert (add (-1, -2) == -3);
}
```

(6) Conditional Compilation
In the D language program no pre-processor is required; this saves the line of code for compiling and saves the time as well as space of run time. Conditional compilation statements are part of the language itself. This gives extra usability to program and at the time of debugging process will take less time. The version statement is a lot like #ifdef in C. If a version identifier is defined, the code under it gets compiled; otherwise, it doesn't [3]

```
version (Linux)
import std.c.linux.linux;
else version (Win32)
import std.windows.windows;
```

(7) Type Inference

With the help of auto key word; D complier can understand the data type automatically. This gives the optimal environment during run time [3]

```
auto q = 1; // here q is integer type
auto t= "hello"; // Here t is character type
```

(8) Exceptions

D uses exceptions for error handling as opposed to error codes. Exception handling mechanism of D resembles the way the operating system handles exceptions in an application. It uses the superior try-catch-finally model, which allows cleanup code to be inserted conveniently in the finally block. For certain cases when they finally block is insufficient, scope statements prove to be quite handy.

(9) Classes

Like any object-oriented language, the D has the ability to create object classes. From simple scripts to large projects, D has the potential to scale with any application's needs. Its focus is to combine the power and high performance of C and C++. D uses a single-inheritance paradigm.

Classes are passed by reference rather than by value, so the programmer doesn't have to worry about treating it like a pointer. Furthermore, there is no -> or :: operator, the. (Dot) operator is used in all situations to access members of structs and classes.

```
class Class_name
{
      int i;
      char [] str;
      void doSomething () {...};
}
```

(10) Functions

In D, there is no inline keyword—the compiler decides which functions to inline, so the programmer doesn't even have to worry about it. Functions can be overloaded—this is to say, two functions with the same name can take different parameters. Function parameters can be either in, out, inout or lazy, with 'in' being the default behavior.

Out parameters are simple outputs:

```
void function (out int i)
{
      I += 3;
 }
void main()
{
      int n = 50;
      writefln (n);
      func (n);
      writefln (n);
}
```

inout parameters are read/write, but no new copy is created:

```
void func (inout int a)
{
      if (a >= 0)
            ..
      else
            ..
}
```

Lazy parameters are computed only when they are needed. . Nested functions in D allow the nesting of functions within other functions same as C, C++ and Java.

(11) Templates

D has a highly flexible template system. For starters, the '!' Operator is used for template instantiation. Here is a simple template example:

```
template TCopy (t)
 {
      void copy (T from, out T to)
      {
            to = from;
      }
}

void main ()
{
      int from = 7;
      int to;
      TCopy! (int).copy (from, to);
}
```

4. PROPOSED ALGORITHM

A program using for, while and do-while loop is implemented in C, Java, C# and D language to calculate how much time is required for execution using the below algorithm.

Algorithm in C language:

```
 % start clock();
 % for (i=0; i<n; i++)
% or while(i<n) i++; //i=0
% or do{}while(i<n) //i=0
%printf(i);
 % end clock ();
% printf(end - start)/CLK_TCK));
```

Algorithm in JAVA language:

```
% startTime=System.currentTimeMillis();
% for (i=0; i<n; i++)
% or while(i<n) i++; //i=0
% or do{}while(i<n) //i=0
% println(i);
% endTime=System.currentTimeMillis();
% println(endTime-startTime);
```

Algorithm in C# language:

```
% Stopwatch stopWatch = new Stopwatch();
 % stopWatch.Start();
% for (i=0; i<n; ++i)
% or while(i<n) i++; //i=0
% or do{}while(i<n) //i=0
% stopWatch.Stop();
 % TimeSpan ts = stopWatch.Elapsed
% Console.WriteLine(elapsedTime);
```

Algorithm in D language:

```
% StopWatch sw;
% sw.start;
% for (i=0; i<n; ++i)
% or while(i<n) i++; //i=0
% or do{}while(i<n) //i=0
% sw.stop;
% sw.peek.msecs.writeln(time);
```

We are using different values of n: 200, 400, 600, 800, 1000, 2000, 3000 and 4000 for the calculation of execution time with single loop among three loop which are for, while and do-while, and result has been shown in Table 3, 4 and 5.

5. RUN TIME ENVIRONMENT

All programs were executed on different compliers and interpreters with 1 GB RAM and Intel(R) Pentium(R) dual CPU E2220 @2.40 GHZ, running under Microsoft Window XP service pack 2 2002.The compilers and interpreters are listed in Table 2.

Table 2: Compilers and interpreters used

Language	Complier or execution Platform
C	Turbo C++ IDE
Java	JDK 5.0.30.7
C#	Microsoft Visual studio 2008
D	DMD2

6. EXECUTION TIME

The efficiency of a program depends upon two factors:

1. Space.

2. Time.

In the modern computer technologies, there is a lot of space available and hence a programmer is free from space chores. So the main factor which decides whether the particular algorithm is best or efficient to perform certain task depends on its execution time. We can define execution time as time in which a particular algorithm generates output from 'n' inputs.

We can calculate the execution time for an algorithm or a program in three cases:

1. Worst Case: In this case program requires maximum time to execute for 'n' inputs.

2. Best Case: Best case occurs when program require minimum time to execute for 'n' inputs.

3. Average Case: Average case occurs when program execute in a time that is between its maximum and minimum time requirement for 'n' inputs [9-10].

7. RESULT

The execution time of a given algorithms is equal to the time spent by the processed algorithms from entry point to exit. Here we have defined and calculated the execution time for the different algorithms. It can depend upon the work-satiation which the stakeholder will use for analyzing the time with different languages. Table [3-5] shows the comparison of execution time in different languages using same program logic i.e. loop implementation (for, while and do-while). Three different loops (for, while and do-while) have been implemented independently for the same value of n in different languages (C, C#, JAVA and D) and the execution time is calculated in sec. The results states that for the given problem; the D language is more productive than most conventional languages, In terms of run time the D language is often better than Java, C and C#.

Table 3: Comparisons of Execution Time using for loop

Value of N	Time in Sec (Using for loop in JAVA)	Time in Sec (Using for loop in C)	Time in Sec (Using for loop in C#)	Time in Sec (Using for loop in D)
0-200	0.078	0.329670	0.06	0.010
0-400	0.125	0.769231	0.10	0.040
0-600	0.172	1.208791	0.11	0.054
0-800	0.234	1.648352	0.12	0.068
0-1000	0.282	2.087912	0.13	0.083
0-2000	0.532	4.178824	0.18	0.156
0-3000	0.782	6.373626	0.24	0.229
0-4000	1.031	8.571429	0.32	0.301

Table 4: Comparisons of Execution Time using while Loop

Value of N	Time in Sec (Using while loop in JAVA)	Time in Sec (Using while loop in C)	Time in Sec (Using while loop in C#)	Time in Sec (Using while loop in D)
0-200	0.078	0.329670	0.06	0.010
0-400	0.125	0.769231	0.09	0.039
0-600	0.141	1.208791	0.11	0.054
0-800	0.250	1.648352	0.12	0.068
0-1000	0.281	2.087912	0.13	0.082
0-2000	0.531	4.285714	0.18	0.155
0-3000	0.782	6.428571	0.24	0.228
0-4000	1.016	8.571429	0.31	0.303

Table 5: Comparisons of Execution Time using do-while Loop

Value of N	Time in Sec (Using do-while loop in JAVA)	Time in Sec (Using do- while loop in C)	Time in Sec (Using do-while loop in C#)	Time in Sec (Using do-while loop in D)
0-200	0.078	0.329670	0.06	0.025
0-400	0.125	0.769231	0.09	0.040
0-600	0.172	1.208791	0.11	0.054
0-800	0.234	1.648352	0.12	0.068
0-1000	0.282	2.087912	0.13	0.082
0-2000	0.532	4.236769	0.18	0.155
0-3000	0.797	6.428571	0.24	0.228
0-4000	1.032	8.626374	0.31	0.301

Graphical representation and some other program with execution time are discussed in Appendix A:

8. CONCLUSIONS

It can be concluded that taking into consideration a system-level language for application execution , D Programming language would serve as a better choice over other platforms .One of the more unique aspects of D language is that it takes less time for the execution of a program. This paper provides same objective information comparing serial languages, namely C, JAVA, C# and D languages. The task of the program is to find out the execution time in different languages. The common software engineering wisdom says that the number of lines coded in the D language per hour is independent of the language platform and this holds fairly well across all languages. We find that D language clearly provides a much faster execution time than C, C# and JAVA. In general the difference between languages does not matter much as compared to the typical differences due to different programmers working on the same language.

Summing up, it appears warranted saying that the execution platform of D language is more productive than that of JAVA, C# and C.

9. REFERENCES

[1] Lutz Prechelt, “An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program”, Technical Report 2000-5, March 10, 2000.

[2] Ali Çehreli, "Programming in D" D.ershane Series, revision: r501, 2012-08-23.2012.

[3] Overview D Programming Language, http://dlang.org/overview.html, Retrieved Jan 2014

[4] Schildt, Herbert, “C++ the Complete Reference”, Third ed, Osborne McGraw-Hill., ISBN 978-0-07-882476-0.

[5] Giannini, Mario; Code Fighter, Inc.; Columbia University (2004). "C/C++". In Hossein Bidgoli. The Internet encyclopedia. 1. John Wiley and Sons. p. 164, ISBN 0-471-22201-1.

[6] Bagley: http://www.bagley.org/~doug/shootout/, Jan 2013.

[7] Hao Chen, "Comparative Study of C, C++, C# and Java Programming Languages", vaasan ammattikorkeakouluuniversity of applied sciences degree program of information technology, 2010.

[8] D programming Language. Arrays: http://dlang.org/arrays.html, Retrieved June 2014

[9] P. Puschner, Ch. Koza, “Calculating the maximum execution time of real-time programs”, Real-Time Systems September 1989, Volume 1, Issue 2, pp 159-176,DOI 10.1007/BF00571421 Print ISSN 0922-6443 Online ISSN 1573-1383, Springer link.

[10] Peter P. Puschner, Anton V. Schedl, “Computing Maximum Task Execution Times -A Graph-Based Approach,Real-Time Systems”, July 1997, Volume 13, Issue 1, pp 67-91,DOI 10.1023/A:1007905003094 Print ISSN 0922-6443 Online ISSN 1573-1383,Springer link.

[11] Patricia K. Lawlis, C.J. kemp, "Guidelines for Choosing a Computer Language: Support for the Visionary Organization". Ada Information Clearinghouse. Retrieved 18 July 2006.

[12] TIOBE Software, http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, Retrieved by Jan 2014

[13] "TIOBE Programming Community Index". 2009.

[14] Stroustrup, Bjarne, "Bjarne Stroustrup's FAQ – When was C++ invented?", Retrieved Jan 2014

[15] C. Brandolese, W. Fornaciari, F. Salice, D. Sciuto, "Source–Level Execution Time Estimation of C Programs", Politecnico di Milano Piazza L. da Vinci, 32 20133 Milano, Italy.

[16] Robert Henderson, Benjamin Zorn, “Comparison of object-oriented programming in four modern languages”, software-practice and experience, vol. 24(11), 1077–1095 (November 1994).

[17] Luiz Fernando Capretz, “A Brief History of the Object-Oriented Approach”, ACM SIGSOFT, Software Engineering Notes vol 28 no 2, March 2003 Page 1.

[18] Nami, Mohammad Reza Hassani, Fatemeh, "A comparative evaluation of the Z, CSP, RSL, and VDM languages", ACM SIGSOFT Software Engineering Notes, Volume 34, issue 3 (May 30, 2009), p. 1-4. ISSN: 0163-5948 DOI: 10.1145/1527202.1527211.

Appendix A:

Program: 1

1000 random number have been generated using same program logic (Explain in algorithm 1) in different language i.e. C, java, C# and D languages [11]. And calculates the how much time is required to run the program in different languages which is shown in table 5.

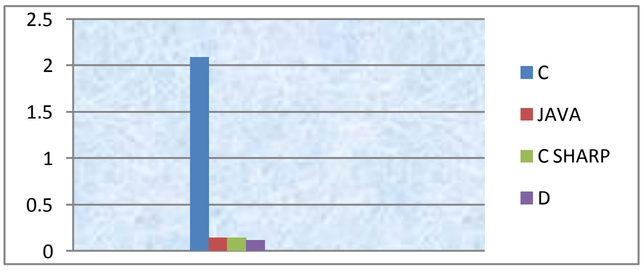
Proposed Algorithm 1:

```
% IM 139968
%  IA 3877
% IC 29573
% last 42
% gen_random(double max) {
%  last = (last * IA + IC) % IM;
%  return( max * last / IM );}
 %  int N =1000;
%   while (N--) {
 %   result = gen_random(100.0);}
 % printf( result);
```

The execution time is shown in Table 6.

Table 6: Execution Time for Random Number Generator Program

Language	Execution Time in Second
C	2.0879120
Java	0.141
C Sharp	0.14
D	0.119



Program 2:

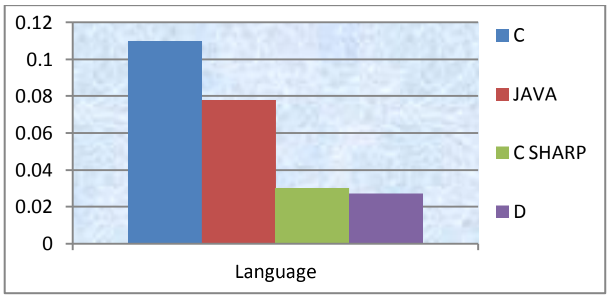
Program has been written (Hybrid Program) in different languages (C, C#, JAVAand D) to print the factorial number starting from 0 to 14, (because after the given range long int limt has been exceed, so we are using said range) and square root of first 45 number (0-44) and first 45 Fibonacci series number (Explain in algorithm 2). And calculate the how much time is required to run the program in different languages which is shown in table 7.

Table 7: Execution Time of Program 2

Languages	Execution Time in Second
C	0.109890
Java	0.078
C #	0.030
D	0.027

Proposed Algorithm 2:

```
 % printf("%ld\n%ld",i,j); //printing first two values.
%  for(k=2;k<r;k++){
 % f=i+j;
 %  i=j;
 %  j=f;
 % printf(" %ld\n",j);}
 % for(w=0;w<=10;w++){
 % l=fact(w);
 % printf(w,l);  }
 %   for(g=0;g<r;g++){
  %  p=sqrt(g); }
```


Language
C
JAVA
C SHARP
D