International Journal of Engineering Applied Sciences and Technology, 2017
Vol. 2, Issue 4, ISSN No. 2455-2143, Pages 114-116
Published Online February-March 2017 in IJEAST (http://www.ijeast.com)

DEGRADATION OF PROCEDURAL LANGUAGES – A REVIEW

Supriya Kumari
Department of Computer Engg.
G.V.I.E.T , Banur, Punjab, India

Amir Shaikh
Department of Marine Engg.
G.V.I.E.T , Banur, Punjab, India

Shaikh Mustufa
Department of Marine Engg.
G.V.I.E.T , Banur, Punjab, India

Abstract - Now a day's C programmers are hard to find. Students and learners are avoiding C programming over C++ and other programming languages. In this research we tried to find out the reasons why programmers are losing interest towards C programmers. We have read many articles of well-known programmers while researching, programming but still they prefer other languages. We too cannot deny that C++ is more developed programming language than C.

Keywords - Lexical, Recursion, Polymorphism, Procedural, Paradigm, Pernicious, Semantics

I. INTRODUCTION

C has withstood the test of time. It's been in use for over 4 decades .C is a general-purpose, imperative computer programming language, supporting structured programming, lexica variable scope and recursion, while a static type system prevents many unintended operations. By design, C provides constructs that map efficiently to typical machine instructions, and therefore it has found lasting use in applications that had formerly been coded in assembly language, including operating systems, as well as various application software for computers ranging from supercomputers to embedded systems. As it is the basic language so all the other language are based on this language.

The origin of C is closely tied to the development of the UNIX operating system, originally implemented in assembly language on a PDP-7 by Dennis Ritchie and Ken Thompson, incorporating several ideas from colleagues. Eventually, they decided to port the operating system to a PDP-11. The original PDP-11 version of UNIX was developed in assembly language. The developers were considering rewriting the system using the B language, Thompson's simplified version of BCPL. However B's inability to take advantage of some of the PDP-11's features, notably byte addressability, led to C. The name of C was chosen simply as the next after B.

The development of C started in 1972 on the PDP-11 UNIX system and first appeared in Version 2 UNIX. The language was not initially designed with portability in mind, but soon ran on different platforms as well: a compiler for the Honeywell 6000 was written within the first year of C's history, while an IBM System/370 port followed soon.

Also in 1972, a large part of UNIX was rewritten in C. By 1973, with the addition of structure types, the C language had become powerful enough that most of the UNIX kernel was now in C.

UNIX was one of the first operating system kernels implemented in a language other than assembly. Earlier instances include the Multiple system which was written in PL/I), and Master Control Program (MCP) for the Burroughs B5000 written in ALGOL in 1961. In around 1977, Ritchie and Stephen C. Johnson made further changes to the language to facilitate portability of the UNIX operating system. Johnson's Portable C Compiler served as the basis for several implementations of C on new platforms like-

1 Early developments
2 K&R C
3 ANSI C and ISO C
4 C99
5 C11
6 Embedded C

II. OVERVIEW

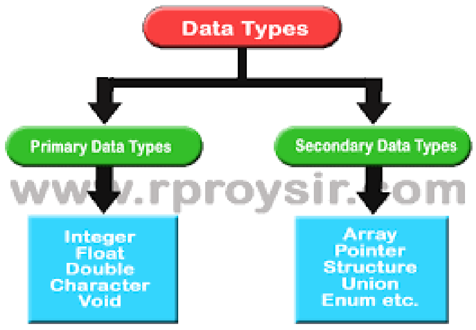
Communicating with a computer involves speaking the language the computer understands, which immediately rules out English as the language of communication with the computer. However, there is a closed analogy between learning English language and learning C language. Learning C is similar and easier. Instead of straight-away learning how to write programs, we must first know what alphabet, numbers and special symbols are used in C, then how using them, constants, variables and keywords are constructed, and finally, how are these combine to form an instruction. Studies of programming can be generally divided into two main categories, those with a software engineering perspective, and those with a psychological/educational perspective. Our review is based on the experience professional programmers, software developers and experts of C. The C programming language developed at AT & T's Bell laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie.

The C programming language, Kernighan, Brian W; Ritchie, Dennis M.(February 1987) in NJ : Prentice Hall was regarded by many to be the authoritative reference of C. He concluded C is simple and basic language. Johnson, S. C.; Ritchie, D. M. (1978) showed the portability of C programs and the UNIX System. He said that it is popular and major part of an operating system and C language is portable. From the review paper on C programming language by NUI GALWAY (14 January 2009), we studied about different data types which are used in C programming language.

III. DATA TYPES

Programming uses different types of data type. Data types are used to identify the type of data and associated operations of handling it. Data types in C are of two types-

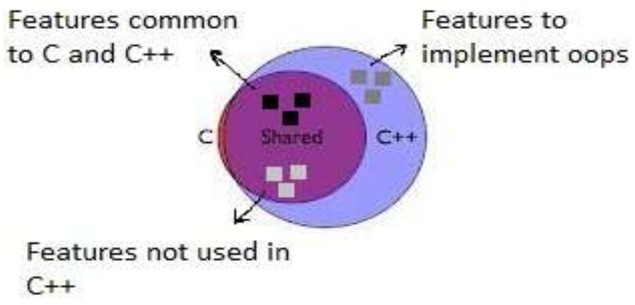
- Primary Data Types – Integer , float, double character , void
- Secondary Data Types- Array , pointer, structure, union , Enum



IV. C IS NOT AN OBJECT ORIENTED PROGRAMMING LANGUAGE

First we need a proper definition of object – oriented programming. A few things that C does not have: It doesn't have any concept of inheritance. While we can embed function pointers in C structs, they aren't proper classes in the OO sense, and such embedded functions pointers don't represent methods. C doesn't have operator or function overloading, which is also a pretty fundamental part of OO method: Polymorphism.

V. WHY PROGRAMMERS CHOOSE C++ OVER C



C is the basic language. The historical relationship between C and C++ has created this problem. C and C++ have now sufficiently diverged that they really ought to be taught as distinct languages, not as "C++ is an improved version of C, and so you can code C in C++". For people who have embraced C++ as the superior language, the persistence of C is irritating - largely because they needlessly conflate the languages.

Unfortunately, in order for an object oriented language to have been widely and quickly adopted, the historical relationship is necessary. At the time C++ was being proposed, C had become a very popular language (especially for people who would never learn structured, high level, compiled language.) Getting THOSE people to advance into object oriented programming required pandering to their affection for C. (And frankly, a hell of a lot of languages STILL look like C - Java, and Perl in particular.) C had a pernicious way of closing the minds of people to alternative languages - and that tendency seems to continue with C++.

VI. DIFFICULTIES A C PROGRAMMER FACES.

- C is Procedural Language.
- No virtual Functions are present in C
- In C, Polymorphism is not possible.
- Operator overloading is not possible in C.
- Top down approach is used in Program Design.
- No namespace Feature is present in C Language.
- Multiple Declarations of global variables are allowed.
- In C
  - scanf() Function used for Input.
  - printf() Function used for output.

- Mapping between Data and Function is difficult and complicated.
- In C, we can call main() Function through other Functions
- C requires all the variables to be defined at the starting of a scope.
- No inheritance is possible in C.
- In C, malloc() and calloc() Functions are used for Memory Allocation and free() function for memory Deal locating.
- It supports built-in and primitive data types.
- In C, Exception Handling is not present.

VII. THE TASK

Learning to program is not easy. C is simple and basic language but the coding of C is too difficult. C language is a computer oriented language. In a good overview what is involved Verilog HDL (2010), describes the importance and difficulties of C language. He believed that nobody can learn C++ or JAVA directly. This is because while learning these languages we have things like classes, objects, inheritance, polymorphism, templates, exception handling, references etc. do deal with apart from knowing the actual language elements. Nick Parlance describes that C's type system and error checks exist only at compile-time. The compiled code runs in a

Stripped down run-time model with no safety checks for bad type casts, bad array indices, or bad pointers. There is no garbage collector to manage memory. Instead the programmer manages heap memory manually. All this makes C fast but fragile.

VIII. ANALYSIS – WHERE C FITS

Because of the above features, C is hard for beginners. This feature can work fine in one

Context but crash in another. The programmer needs to understand how the features work and use them correctly. On the other hand, the number of features is pretty small. Like most programmers, I have had some moments of real loathing for the C language. It can be irritatingly obedient -- you type something incorrectly, and it has a way of compiling fine and just doing something you don't expect at run-time. However, as I have become a more experienced C programmer, I have grown to appreciate C's straight-to-the point style. I have learned not to fall into its little traps, and I appreciate its simplicity. Perhaps the best advice is just to be careful. Don't type things in you don't understand. Debugging takes too much time. Have a mental picture (or a real drawing) of how your C code is using memory. That's good advice in any language, but in C it's critical. Perl and Java are more "portable" than C (you can run them on different computers without a recompile). Java and C++ are more structured than C. Structure is useful for large projects. C works best for small projects where performance is important and the programmers have the time and skill to make it work in C. In any case, C is a very popular and influential language. This is mainly because of C's clean (if minimal) style, it's lack of annoying or regrettable constructs, and the relative ease of writing a C compiler.

IX. RESOLUTION

In older times the efficiency, and predictability of performance, were paramount. C was a simple language that was suited for a system programming e.g., it can be used to write an operating system, a driver, or just anything. C language has a procedural approach which makes it a vast and long while C++ is OOPs which makes it easier to understand and the functions present in it are classified into objects which is a real life entity. C is widely criticized as it is also used to argue that C is not good teaching language C aficionados love this aspect of c language because it means that C does not try to protect themselves when they know what they are doing , even if it's risky or obscure they can do it . This is the aspect of language which it's fairly pointless to complaint about.

X. REFERENCE

1) Kernighan, Brian W.; Ritchie, Dennis M. (February 1987). The C Programming Language (1st ed.). Englewood Cliffs, NJ: Prentice Hall. ISBN 0-13-110163-3. Regarded by many to be the authoritative reference on C.
2) Ritchie (1993): "Thompson had made a brief attempt to produce a system coded in an early version of C-before structures – in1972, but gave up the effort.
3) Ritchie (1993): "The scheme of type composition adopted by C owes considerable dept. to Algol 68, although it did not, perhaps, emerge in a form that Algol's adherent would approve of."
4) "Verilog HDL (and C). The Research School of Computer Science at the Australian National University. 2010-06-03. Retrieved 2013-08-19. "1980s: ; Verilog first introduced ; Verilog inspired by the C programming language"
5) "Programming Language Popularity". 2009. Retrieved 16th January, 2009.
6) Johnson, S. C.; Ritchie, D.M. (1978). "Portability of C Programs and the UNIX System".
7) Review on C language – NUI GALWAY, research paper , 14 January 2009, page – 14, 17.
8) Essential C , Nick Parlante (1996-2003) , page – 2 , 3
9) Book Let us C – Yashavant Kanetkar