FW17231

The Development and Influence of CPL

Initially developed in 1963 by researchers at Cambridge University Mathematical Laboratory, later collaborating with the University of London Computer Unit, the Combined Programming Language (CPL) was created so that all types of problem could be solved[1]. It was based off a previous language ALGOL 60, adding features to it such; non-numerical item manipulation, command structures, and comprehensive input-output facilities to name a few[1]. Both the Cambridge Mathematics Department and the London Computer Unit implemented the language on the computer Titan and Atlas respectively[1]. Similar to languages of today, CPL programs are comprised of data structures, functions, and commands[1]; implying the influence that CPL had on programming languages that came after it.

Since CPL could manipulate non-numerical items, it paved the way to the development of a powerful type of processor called a macrogenerator. Processors of this type are used for manipulating symbolic strings, with its input and output being composed of said strings[2]. Operating by a form of substitution, a macrogenerator would produce several lines of output string from a single line of input string, according to what parameters were found in the input string[2]. The rules that determine what to do with the domain strings is a program itself, meaning that it is not tied to a particular assembly program[2]. Macrogenerators can be used on source code to progressively transform them into machine code instructions. CPL utilises macrogenerators in this way so that it can write and define its own compiler in the CPL language[2][3]. This technique is called bootstrapping.

Succeeding CPL was Basic CPL, which was developed by Martin Richards at MIT. BCPL is a simplified version of CPL, created as a tool mainly for writing compilers[4]. The simplified syntax and single data type (binary bit patterns) of BCPL allowed it to be very efficient and easy to compile[4]. Furthermore, Basic CPL did away with the use of the General Purpose Macrogenerator, and instead implemented a sequence of integers in mnemonic form called OCODE[5]. Using OCODE instead of a General Purpose Macrogenerator meant shorter compiling times[5], making BCPL a much more efficient language in terms of both code complexity and compiling time. These main differences are what distinguish the two languages from each other, as in most other aspects they are extremely similar. The simplicity of BCPL meant that it outlived CPL, and was used a great deal more in institutions such as; MIT, Xerox and Cambridge[5], and is still used commercially today[5].

Despite being short lived, there is no denying that CPL was one of the most influential programming languages, maybe ever. From progressing the processing power and usage of macrogenerators, to influencing languages that came after it such as; BCPL, B and even C[5]. Because of the effort put into developing CPL and it compiler, we now have one of the most used languages, C, because of it, and consequently the UNIX operating system too. There is no doubt that the Combined Programming Language was influential, and arguably very successful regarding this.

Bibliography

[1] Barron, D. W. et al., 1963. The main features of CPL. *The Computer Journal,* 6(2), p. 134.

[2] Strachey, C., 1965. A general purpose macrogenerator. *The Computer Journal,* 8(3), p. 225.

[3] Buth, B. et al., 1992. Provably Correct Compiler Development and Implementation. *Lecture Notes in Computer Science,* Volume 641, pp. 141-155.

[4] Richards, M., 1969. BCPL: a Tool for Compiler Writing and System Programming. *Spring Joint Computer Conference,* p. 557.

[5] Richards, M., 2012. How BCPL evolved from CPL. *The Computer Journal,* 56(5), pp. 664-670.