This history was written in the spring of 2000 when I was in eighth grade. For several years after I wrote it, the text was available online and it became a reference for other articles, Wikipedia entries, and even college courses. I have placed the report here for posterity and amusement (how could I have possibly left out Python??). In 2004, I made two updates in response to e-mailed criticisms, but the text is otherwise unchanged. For a more up-to-date reference, I shamelessly point you to Wikipedia.
— Andrew Ferguson

A History of Computer Programming Languages

Ever since the invention of Charles Babbage's difference engine in 1822, computers have required a means of instructing them to perform a specific task. This means is known as a programming language. Computer languages were first composed of a series of steps to wire a particular program; these morphed into a series of steps keyed into the computer and then executed; later these languages acquired advanced features such as logical branching and object orientation. The computer languages of the last fifty years have come in two stages, the first major languages and the second major languages, which are in use today.

In the beginning, Charles Babbage's difference engine could only be made to execute tasks by changing the gears which executed the calculations. Thus, the earliest form of a computer language was physical motion. Eventually, physical motion was replaced by electrical signals when the US Government built the ENIAC in 1942. It followed many of the same principles of Babbage's engine and hence, could only be "programmed" by presetting switches and rewiring the entire system for each new "program" or calculation. This process proved to be very tedious.

In 1945, John Von Neumann was working at the Institute for Advanced Study. He developed two important concepts that directly affected the path of computer programming languages. The first was known as "shared-program technique" (www.softlord.com). This technique stated that the actual computer hardware should be simple and not need to be hand-wired for each program. Instead, complex instructions should be used to control the simple hardware, allowing it to be reprogrammed much faster.

The second concept was also extremely important to the development of programming languages. Von Neumann called it "conditional control transfer" (www.softlord.com). This idea gave rise to the notion of subroutines, or small blocks of code that could be jumped to in any order, instead of a single set of chronologically ordered steps for the computer to take. The second part of the idea stated that computer code should be able to branch based on logical statements such as IF (expression) THEN, and looped such as with a FOR statement.

"Conditional control transfer" gave rise to the idea of "libraries," which are blocks of code that can be reused over and over. *(Updated Aug 1 2004: Around this time, Konrad Zuse, a German, was inventing his own computing systems independently and developed many of the same concepts, both in his machines and in the Plankalkul programming language. Alas, his work did not become widely known until much later. For more information, see this website: http://www.epemag.com/zuse/, or the entries on Wikipedia: Konrad Zuse and Plankalkul.)*

In 1949, a few years after Von Neumann's work, the language Short Code appeared (www.byte.com). It was the first computer language for electronic devices and it required the programmer to change its statements into 0's and 1's by hand. Still, it was the first step towards the complex languages of today. In 1951, Grace Hopper wrote the first compiler, A-0 (www.byte.com). A compiler is a program that turns the language's statements into 0's and 1's for the computer to understand. This lead to faster programming, as the programmer no longer had to do the work by hand.

In 1957, the first of the major languages appeared in the form of FORTRAN. Its name stands for FORmula TRANslating system. The language was designed at IBM for scientific computing. The components were very simple, and provided the programmer with low-level access to the computers innards. Today, this language would be considered restrictive as it only included IF, DO, and GOTO statements, but at the time, these commands were a big step forward. The basic types of data in use today got their start in FORTRAN, these included logical variables (TRUE or FALSE), and integer, real, and double-precision numbers.

Though FORTAN was good at handling numbers, it was not so good at handling input and output, which mattered most to business computing. Business computing started to take off in 1959, and because of this, COBOL was developed. It was designed from the ground up as the language for businessmen. Its only data types were numbers and strings of text. It also allowed for these to be grouped into arrays and records, so that data could be tracked and organized better. It is interesting to note that a COBOL program is built in a way similar to an essay, with four or five major sections that build into an elegant whole. COBOL statements also have a very English-like grammar, making it quite easy to learn. All of these features were designed to make it easier for the average business to learn and adopt it.

(Updated Aug 11 2004) In 1958, John McCarthy of MIT created the LISt Processing (or LISP) language. It was designed for Artificial Intelligence (AI) research. Because it was designed for a specialized field, the original release of LISP had a unique syntax: essentially none. Programmers wrote code in parse trees, which are usually a compiler-generated intermediary between higher syntax (such as in C or Java) and lower-level code. Another obvious difference between this language (in original form) and other languages is that the basic and only type of data is the list; in the mid-1960's, LISP acquired other data types. A LISP list is denoted by a sequence of items enclosed by parentheses. LISP programs themselves are written as a set of lists, so that LISP has the unique ability to modify itself, and hence grow on its own. The LISP syntax was known as "Cambridge Polish," as it was very different from standard Boolean logic (Wexelblat, 177):

x V y - Cambridge Polish, what was used to describe the LISP

program
OR(x,y) - parenthesized prefix notation, what was used in the LISP program
x OR y - standard Boolean logic

LISP remains in use today because its highly specialized and abstract nature.

The Algol language was created by a committee for scientific use in 1958. It's major contribution is being the root of the tree that has led to such languages as Pascal, C, C++, and Java. It was also the first language with a formal grammar, known as Backus-Naar Form or BNF (*McGraw-Hill Encyclopedia of Science and Technology*, 454). Though Algol implemented some novel concepts, such as recursive calling of functions, the next version of the language, Algol 68, became bloated and difficult to use (www.byte.com). This lead to the adoption of smaller and more compact languages, such as Pascal.

Pascal was begun in 1968 by Niklaus Wirth. Its development was mainly out of necessity for a good teaching tool. In the beginning, the language designers had no hopes for it to enjoy widespread adoption. Instead, they concentrated on developing good tools for teaching such as a debugger and editing system and support for common early microprocessor machines which were in use in teaching institutions.

Pascal was designed in a very orderly approach, it combined many of the best features of the languages in use at the time, COBOL, FORTRAN, and ALGOL. While doing so, many of the irregularities and oddball statements of these languages were cleaned up, which helped it gain users (Bergin, 100-101). The combination of features, input/output *and* solid mathematical features, made it a highly successful language. Pascal also improved the "pointer" data type, a very powerful feature of any language that implements it. It also added a CASE statement, that allowed instructions to to branch like a tree in such a manner:

```
CASE expression OF
   possible-expression-value-1:
       statements to execute...
   possible-expression-value-2:
       statements to execute...
END
```

Pascal also helped the development of dynamic variables, which could be created while a program was being run, through the NEW and DISPOSE commands. However, Pascal did not implement dynamic arrays, or groups of variables, which proved to be needed and led to its downfall (Bergin, 101-102). Wirth later created a successor to Pascal, Modula-2, but by the time it appeared, C was gaining popularity and users at a rapid pace.

C was developed in 1972 by Dennis Ritchie while working at Bell Labs in New Jersey. The transition in usage from the first major languages to the major languages of today occurred with the transition between Pascal and C. Its direct ancestors are B and BCPL, but its similarities to Pascal are quite obvious. All of the features of Pascal, including the new ones such as the CASE statement are available in C. C uses pointers extensively and was built to be fast and powerful at the expense of being hard to read. But because it fixed most of the mistakes Pascal had, it won over former-Pascal users quite rapidly.

Ritchie developed C for the new Unix system being created at the same time. Because of this, C and Unix go hand in hand. Unix gives C such advanced features as dynamic variables, multitasking, interrupt handling, forking, and strong, low-level, input-output. Because of this, C is very commonly used to program operating systems such as Unix, Windows, the MacOS, and Linux.

In the late 1970's and early 1980's, a new programing method was being developed. It was known as Object Oriented Programming, or OOP. Objects are pieces of data that can be packaged and manipulated by the programmer. Bjarne Stroustroup liked this method and developed extensions to C known as "C With Classes." This set of extensions developed into the full-featured language C++, which was released in 1983.

C++ was designed to organize the raw power of C using OOP, but maintain the speed of C and be able to run on many different types of computers. C++ is most often used in simulations, such as games. C++ provides an elegant way to track and manipulate hundreds of instances of people in elevators, or armies filled with different types of soldiers. It is the language of choice in today's AP Computer Science courses.

In the early 1990's, interactive TV was the technology of the future. Sun Microsystems decided that interactive TV needed a special, portable (can run on many types of machines), language. This language eventually became Java. In 1994, the Java project team changed their focus to the web, which was becoming "the cool thing" after interactive TV failed. The next year, Netscape licensed Java for use in their internet browser, Navigator. At this point, Java became the language of the future and several companies announced applications which would be written in Java, none of which came into use.

Though Java has very lofty goals and is a text-book example of a good language, it may be the "language that wasn't." It has serious optimization problems, meaning that programs written in it run very slowly. And Sun has hurt Java's acceptance by engaging in political battles over it with Microsoft. But Java may wind up as the instructional language of tomorrow as it is truly object-oriented and implements advanced techniques such as true portability of code and garbage collection.

Visual Basic is often taught as a first programming language today as it is based on the BASIC language developed in 1964 by John Kemeny and Thomas Kurtz. BASIC is a very limited language and was designed for non-computer science people. Statements are chiefly run sequentially, but program control can change based on IF..THEN, and GOSUB statements which execute a certain block of code and then return to the original point in the program's flow.

Microsoft has extended BASIC in its Visual Basic (VB) product. The heart of VB is the form, or blank window on which you drag and drop components such as menus, pictures, and slider bars. These items are known as "widgets." Widgets have properties (such as its color) and events (such as clicks and double-clicks) and are central to building any user interface today in any language. VB is most often used today to create quick and simple interfaces to other Microsoft products such as Excel and Access without needing a lot of code, though it is possible to create full applications with it.

Perl has often been described as the “duct tape of the Internet,” because it is most often used as the engine for a web interface or in scripts that modify configuration files. It has very strong text matching functions which make it ideal for these tasks. Perl was developed by Larry Wall in 1987 because the Unix sed and awk tools (used for text manipulation) were no longer strong enough to support his needs. Depending on whom you ask, Perl stands for Practical Extraction and Reporting Language or Pathologically Eclectic Rubbish Lister.

Programming languages have been under development for years and will remain so for many years to come. They got their start with a list of steps to wire a computer to perform a task. These steps eventually found their way into software and began to acquire newer and better features. The first major languages were characterized by the simple fact that they were intended for one purpose and one purpose only, while the languages of today are differentiated by the way they are programmed in, as they can be used for almost any purpose. And perhaps the languages of tomorrow will be more natural with the invention of quantum and biological computers.

Bibliography

- “A Brief History of Programming Languages.” http://www.byte.com/art/9509/se7/artl9.htm. Cited, March 25, 2000.
- “A Short History of the Computer.” http://www.softlord.com/comp/. Jeremy Myers. Cited, March 25, 2000.
- Bergin, Thomas J. and Richard G. Gibson, eds. *History of Programming Languages-II*. New York: ACM Press, 1996.
- Christiansen, Tom and Nathan Torkington. *Perlfaq1 Unix Manpage*. Perl 5 Porters, 1997-1999.
- Christiansen, Tom and Nathan Torkington. *Perlhist Unix Manpage*. Perl 5 Porters, 1997-1999.
- “Java History.” http://ils.unc.edu/blaze/java/javahist.html. Cited, March 29, 2000.
- “Programming Languages.” *McGraw-Hill Encyclopedia of Science and Technology*. New York: McGraw-Hill, 1997.
- Wexelblat, Richard L., ed. *History of Programming Languages*. New York: Academic Press, 1981.