

A short history of programming languages

Sebastian Gosch

Computers and the languages that control them are all around us. In this work i'll try to outline how we got here.

1 How computers and programming got started

1.1 Babbage's Analytical Machine

Preceded by the chinese abacus and some other calculating devices, such as the Pascaline built by Blaise Pascal and the Difference Engine by himself, Charles Babbage proposed in 1837 a fully programmable machine he called the Analytical Machine. He worked on it until his death in 1871, but never managed to build one.

The Analytical Machine was astonishingly similar to modern-day computers, in that it stored it's program data and input data seperatly, had a central processing unit and a printer output data. Theoretically, if built, it would have been Touring-complete.

Ada Augusta King, Countess of Lovelace, then wrote programmes for the machine, including one to compute Bernoulli-Numbers. In these programmes she used programming concepts now ubiquitous in modern programming, such as subroutines, as a reusable set of instructions, loops and conditional jumps.

1.2 John Von Neuman

Much later, around 1945, John von Neuman formalized these ideas into two concepts. One, the "shared-program technique", stating that a computer should not need changes done to its hardware to run a different programme, but rather be freely programmable. The other, "conditional control transfer", formalising the idea that code can be put in small blocks which do not have to be computed sequentially but can be accessed according to conditions in the code.

So now, after the ENIAC, which could only run one specific programme and needed massive hardware reconfiguration to run another, several computers, such as the ORDVAC, EDVAC, were being built based more or less on von Neumans ideas. These machines were programmed by feeding them zeros and ones on punching cards.

1.3 The first programming language and the first compiler

The first programming language that was not machine code, was Short Code in 1949. It provided basic arithmetic and allowed for branching and calls to a library of functions, but had to be compiled into bits by hand and was about 50 times slower than machine code.

In 1951 Grace Hopper programmed the first compiler A-0, paving the way for the higher level programming languages used today.

2 Significance and characteristics of higher-level programming

Higher level programming lets the programmer concentrate on solving a problem without having to worry too much about a computer's hardware. A drawback though is the slower computing time, as the code will first be translated into machine code and then be run.

2.1 FORTRAN - 1957

The first big language was Fortran (FORmula TRANslation) which was mainly intended and used for scientific computing, the military projects and later the Space Program being prominent examples. It was the first language to introduce data types, such as boolean, integer, real and double-precision numbers. It spawned a whole lot of dialects which adapted the language and kept it up to date in the course of the years and is widely used up until today.

2.2 COBOL - 1959

Whilst Fortran was great for number crunching it did lack a proper I/O, and in 1959 the Conference on Data Systems and Languages (CODASYL) got together and agreed on Cobol as a language for the business market. It only recognised numbers and strings of text, which were grouped together into arrays providing for good data handling.

Although widely criticized for being just a short term solution to problems perceived in the 1950's and significantly better languages existing on the market today, the language is still in widespread use, due to legacy systems having been dragged along from the very beginning.

2.3 ALGOL - 1958

Implemented in a formal grammar, the Backus-Naur-Form, Algol is the ancestor of pretty much every programming language in use today, in that it is the first block-structured language. Originally intended for scientific computing it soon got replaced by the languages it spawned.

2.4 Lisp - 1958

Lisp is short for List Processing and is quite different from the other languages of that time, or any time even, in that it relies heavily on linked lists as the main data structure. Even the source code itself is made up of lists, thus making the language easy to manipulate.

Lisp is closely linked to AI research.

3 Software crisis and beyond

Up until now all the computing was done on huge computers taking up whole rooms in universities and big companies, like the Univacs or later the PDP series. Personal computers turned up around 1980 bringing new interests and needs.

3.1 Software crisis

As hardware power was growing exponentially, following Moore's Law, software programmers struggled to keep up. More and more lines of code were being written for increasingly complex projects. This led to the software crisis of the 1960's and early 1970's when more and more software projects began to run over-budget and over-time, as well as being of poor quality.

In 1968 Friedrich Bauer addresses the problem at a NATO Conference in 1968 in Garmisch and calls for software engineering, much similar to project management in classical engineering fields. The most interesting proposal from a programming language point of view being modularization of code for easier maintenance, since only parts of the software have to be rewritten, if the need arises. Also modularization encourages and supports collaboration between programmers on software projects.

These concepts have found their way into today's languages, which pretty much all support modularization in some form or other.

3.2 Object Orientation

Object-oriented programming dates back to the 1960's but only got popular in the early 1990's. It is a special type of modularization intended to ensure a programme's quality. Object-oriented programming languages share a few common concepts which would define them:

- Class: a class defines the traits shared by all the objects which can be instances of that class
- Object: an object is an instance of such a class
- Method: a method is the ability of such an object

Other concepts in OOP include message passing, inheritance and encapsulation.

3.3 Home Computers

Home computers came into use in the late 1970's, allowing loads of people, who would else would have never seen a computer, to programme on their own. This led to the rise of languages otherwise not notable such as Pascal, which was intended as an educational tool, with very strict syntax but gathered widespread adoption outside expectations, and the keeping of Basic which's use was declining, but which spread again after versions of Basic were distributed with pretty much every home computer.

4 Languages Today

Nowadays thousands of programming languages are being used, some of the first ones still, dialects of the early languages, newer languages designed with the knowledge of the last 50 years, or even completely new languages are still being proposed and implemented. The world is an ever changing place with new problems, ideas and hardware environments sprouting up every other second.

Choosing the best for a given software project, would depend on the scope of the project, the inclination of the programmer, the existence of legacy software which would have to be considered, as well as a whole other load of reasons. I will now try and detail some of the better-known languages of today.

4.1 C

C used a lot of concepts known from - but is not a successor of - Pascal, although a lot harder to read and more effective in terms of computing speed. It was mainly associated, but not limited to, the UNIX operating system creating a synergy effect in acceptance, helping both projects greatly. Since its inception it has been it has had a range of successors such as C with Classes in 1980 and later, based on that, C++ which implemented the ideas of OOP and made it compatible for today.

4.2 Java

Designed by Sun Microsystems in 1990 and intended to be very portable, by not allowing dialects and being compiled into bytecode which then runs on a Java Virtual Machine. Java supports a whole load of modern programming concepts but omits others, such as pointers, in favor of security Java has got a range of shortcomings, such as its low speed and its C-like code which is not very readable.

4.3 Domainspecific languages

In the course of ongoing specializations in the buisness and computing world programming languages have reflected this trend by Nowadays, almost as often as a general-purpose language, a domainspecific language will be used to solve a little problem. Often these languages will be used side by side in a software project. Such languages include database-query languages, like sql, graphical and printing languages, such as Postscript, scripting languages such as PHP and many more.

5 Bibliography

- **Klaus Bothe** - *Von Algol nach Java - Kopntinuit und Wandel in Programmiersprachen* - 2006

- *The Language Center* - <http://www.engin.umd.umich.edu/CIS/course.des/cis400/index.html> accessed on 25.05.07

- *The history of programming languages* - <http://www2.lv.psu.edu/ojj/courses/ist-240/reports/spring2001/faculty-cb-bc-kf/historyindex.html> accessed on 25.05.07

- *pretty much all of Wikipedia* - <http://en.wikipedia.com> accessed 25.05.07