Introduction to Programming languages

A computer is really a collection of millions of electric switches. These switches, which are microscopic and have no moving parts, can be turned ON or OFF and doing so, control tiny currents of electricity. Patterns of switches can be made to represent, manipulate and store data. A switch in the OFF state can be represented by the number 0, while a switch in the ON state is represented by a 1. From the most exciting movie on DVD to the simplest computer application, all are nothing more than groups of switches turning on and off at extremely high speed.  In a typical computer, the processor and the electronic memory are the main locations where you will find these devices.

**
Figure Switch**

The very first [Intel](http://www.intel.com) microprocessor introduced in 1971 named the 4004, contained 2300 transistors.  A transistor is a tiny silicon device that can act as an on off switch.  By comparison, the latest Pentium processor contains 410 million transistors. (When we buy a computer we are usually most concerned with the amount of memory, in megabytes, and its speed in gigahertz.  This really translates into how many switches and how fast they can operate.)

So, how do you get these numerous, tiny, and non moving switches to turn on and off. Without revealing the whole story it is sufficient to say that we use an application called a **programming language**.

A program is used by people to instruct computers how to perform some task. Like human languages, there have been numerous and varied programming languages. The first language was little more than a series of wiring or switch settings which had to be redone every time the computer was needed to do something different. In the 1950's, [Grace Hooper](http://www.agnesscott.edu/lriddle/women/hopper.htm), designed the first language that could be written in English. It was called Flow-Matic.

This was quickly followed by FORTRAN ([Formula Translator](http://www.ibiblio.org/pub/languages/fortran/ch1-1.html)), a high level language (HLL) developed for science and engineering applications.  The concept was to create the program using easily understood English commands where the programmer needed no knowledge of how the computer hardware operated.  The high level code was then converted into machine code instructions by an application called a **compiler**.  These instructions could then control the computer in doing some task.  The major drawback with early FORTRAN was the use of punched cards. These were cardboard cards with holes punched out to represent the program and the data it was to manipulate.  Editing an error meant re-punching the cards and then placing them in the correct order.  However, it was a definite improvement over programming in difficult machine language and opened the use of computers to non experts.

**
Figure Early IBM card punch machine and punch card**

In 1964 Thomas Kurtz and John Kemeny, mathematics professors at Dartmouth College in New Hampshire, USA, created a language which was easy for anyone to understand and use. They called it **BASIC** and in doing so unknowingly laid the basis for personal computing.  When small, inexpensive computers appeared in the 1970’s, a small company called Microsoft saw that personal computers would need some means of instruction and developed a version of Basic that was the direct ancestor of Visual Basic used today. BASIC is short for Beginners' All-purpose Symbolic Instruction Code.

The original BASIC language used easy to remember commands written in English and each line of code started with a line number. The underlying program, called an **interpreter**, translated these instructions into the 0’s and 1’s (called machine language or binary code) that were used to set the switch patterns that make up the computer's memory. The high level BASIC code was "interpreted" every time the program was executed which meant that it wasn't as fast as code that was compiled (as in FORTRAN).  However, this was made up for in the ease of editing, especially when computer monitors became available and, because the program didn't have to be re-compiled. A sample of an early BASIC program would look like:

**10 Let A = 5
20 Let B = 10
30 Let C = A + B
40 Print “The sum of “; A ;” and “; B; “ is “;C
50 END**

When this was typed into a computer running BASIC (the first basic computers used [teletype machines](http://www.columbia.edu/acis/history/teletype.html) as input devices) and the command **RUN** was issued, the result would be displayed as:

**The sum of 5 and 10 is 15**
This type of programming has been described as linear (also known as procedural). When the RUN command is issued, the computer will step through each instruction until the end is reached. In most text based BASIC languages, graphics were primitive and difficult to create.  The last version of Microsoft BASIC developed was Quick Basic 4.5.  A variation called QBasic was shipped with early versions of the Windows operating system up to and including Windows 98.

**
Figure QBasic. Note line numbers not used**

Another illustration of how a high level language gave to anyone interested the ability to program a computer can be given when we contrast it with a language called assembler.  Assembly language is very close to machine language where mnemonics (shortened names) represented the functions of the computer hardware.  A good knowledge of how the machine operates at the most basic level is needed.  For example, the following program displays two words on the screen:

|  |  |
| --- | --- |
| **READONLYADR R0, text SWI "OS\_PrettyPrint"SWI "OS\_NewLine"MOV PC, R14 .text EQUS "Hey Eric"EQUB 0ALIGN** | **GET h.SWIsAREA |asm$$code|, CODE,ENTRYADR r0, textSWI OS\_PrettyPrintSWI OS\_NewLineMOV pc, r14text DCB "Hey Eric", 0ALIGN** |

Written in BASIC, the same program would like like:

**Print "Hey Eric"**

Early personal computers using BASIC as the operating system had to be accessed by typing commands. (You can still do this on the latest versions of Windows but most people find the graphical interface much easier). With the advent of the Windows graphical environment in the 1990’s, Microsoft BASIC evolved into Visual Basic.

Visual Basic is an event driven language that uses objects drawn on a form. Event driven means that, unlike the original Basic, the code does not get executed until some event occurs such as clicking a button or dragging a scroll bar. That is, code is linked to objects and to events that happen to those objects. Today there are many powerful languages available for microcomputers so why use Visual Basic? First you can design great looking forms that have a specific function.  Many of the shareware type applications are actually written in Visual Basic. Second, the code used is still a version of BASIC which is easy to use and understand. Although we use the term easy, recent versions of Visual Basic are very powerful and sophisticated.