



AMUG Tenderfoots #5

Introduction to Programming?

By L. Frank Turovich

Copyright ©1991, All Rights Reserved.

What is Programming?

Programming is a method of telling the computer exactly what we want it to do. To do this we generate a series of explicit instructions that tell the computer in a step-by-step format exactly what we want it to perform. In a nutshell, that is all programming amounts to, creating a list of instructions that tells the computer what its supposed to do. Each step along the way must be detailed exactly in order for the program to properly execute the step. If the step is ill-defined, ambiguous, or incorrect, the program simply fails to operate correctly.

Definitions

Before we can really get started we need to examine a few definitions so that we all start out on the same footing. Source code -- the actual program code as implemented in the language of choice. This is the raw instructional text that will be turned into a program or application when complete.

Program -- the final code as it executes. Sometimes a stand-alone program, but can be tied to the programming language as in HyperCard or BASIC.

Application -- the final code format ready for distribution,

Language -- used to create the source code of a program. In the end, all languages do the same thing, run a program on a computer, but their methods to get there are sometimes different. C, Pascal, and BASIC are some example languages.

Macros are Programs Too

If you regularly use any major spreadsheet, database, or telecommunications program, or own a macro utility like QuickKeys or Tempo, you have probably already done some programming in the form of macros. Macros are simply mini-programs that perform a specific action upon user request.

Constructing a macro is very similar to writing a program. Each step along the way must be described precisely in order for the macro to execute. It is this attention to detail that makes a macro operate correctly. Programming simply carries this attention to detail onto a much larger scale.

Choosing a Language

Once you've decided to become a programmer the next decision to make involves deciding on which language to use.

Binary

Originally there was binary language. Binary language is the lowest level of communicating with a computer. It involves real nuts and bolts programming since it deals with the computer as a series of binary numbers. Each binary value represents a command the computer can understand. This binary representation is the real language of computers.

For example, the binary command 01010101 could tell the computer to add two values stored in two different memory locations. The command 10101010 could mean to subtract one value from another. This was how programmers originally spoke to computers. As you can see, it would be very tedious, time-consuming, and difficult to remember one command from another if you did this for very long.

Assembly

One smart programmer got the idea of making the computer do the translating for us. He wrote some routines that allowed him to enter simple three letter commands which the computer then translated into its native binary code. All other programming languages use this very same idea to talk with a computer.

Now, instead of remembering a series of binary numbers, the programmer could use words like ADD or SUB to perform specific commands. This made programming much easier. Even though other languages have appeared assembly allows for the smallest and fastest code production in most situations. It is very detail orientated, requiring some real discipline from the programmer in order to use efficiently.

FORTRAN & COBOL

Other early languages some appeared that specialized their programming methods for particular purposes.

FORTRAN (FORmula TRANslator) excels in executing scientific applications that require heavy mathematical management. COBOL (COmmon Business Oriented Language) has been used extensively for business computing specifically for payroll and accounting purposes.

There application specialties are vast within specific fields, but for general programming uses they lacked many useful and necessary commands or functions.

BASIC

BASIC (Beginners All-purpose Symbolic Instruction Code) has the reputation as being the *lingua franca* of

computer languages. Since its creation it has been adapted to run on nearly all computers in operation today. It spread rapidly through the world, since in many cases, it came bundled with new computers. As such, it was the first programming language many people came in contact with and learned.

BASIC is a rich programming language. Rich, because it has expanded and adapted the best features of other languages as its own, and integrated them into its own easy-to-learn format. Graphics and file commands are numerous, while its string handling capabilities are very extensive. It is a general all-purpose language that can satisfy many programming requirements.

During its beginning BASIC became notorious for creating unstructured programs and suffered from being considered a toy language. Modern BASIC's have overcome those limitations and have become just as highly structured and powerful as other high-level languages, and its much easier to use. While BASIC is not advocated by many "serious" programmers who only know its past reputation, BASIC is still used daily by many major and minor companies to create many very useful and dynamic programs.

Pascal

The Pascal programming language was invented by Nicklaus Wirth. It was originally designed to teach programming students a structure behind which they could write more error free programs. To this end it was much more highly structured than early BASICs.

Pascal is easy to learn, but has strict structuring rules that require the programmer to pre-plan a program before writing it. Pascal also has many built in commands and data structures that make program development swift and practically error free. You can still make logical programming errors, but not with the program structure or the language itself, only in the program's execution.

Pascal was the language of choice by the Macintosh development team and has been used ever since to build its many functions and commands.

C

C was developed for Bell Labs. It is the current hot language used by many developers and companies. C most nearly straddles the line between Pascal and Assembly language. It is highly structured in program format with an abbreviated and somewhat cryptic command language, but it offers much more control over a computer's microprocessor than Pascal, but is easier to use than Assembly.

HyperTalk

The latest rage in the Macintosh community is HyperTalk, the programming language that makes HyperCard work. HyperTalk is easy to learn, with very English-like commands that make them easy to remember. Its powerful command set includes many commands that make handling complex Macintosh structures like windows and menus very easy to implement. It does suffer from a lack of speed and raw programming power except through external code additions which are normally written in Pascal, C, or BASIC.

HyperTalk is to the 80 and 90's what BASIC was to the 60's and 70's. An easy to learn, simple to implement

language that addresses the programming needs of many beginning programmers.

Why Program?

Programming offers many rewards.

For many people using a computer is chore, something to be dreaded. Part of that is the old assumption that only "smart" people used computers. In the early days it was true that it took a lot of study and dedication to program a computer. That isn't true anymore though, especially when the computer language does most of the work for you. Learning to program will teach you that the computer is simply a tool, like a hammer, or a calculator. It has its uses and it can produce meaningful results, if you know what you are doing.

It is mentally stimulating. Like a puzzle or a murder mystery, creating a computer program that does exactly what you want is very exciting. Wrestling programming bugs into submission to execute your commands correctly is both challenging and very learning oriented. I learn something every time I program, about the computer, the language, and even myself.

In the next couple of months we will look at programming on the Macintosh. Along the way I hope to show you that it isn't as hard to do as some people think, nor is it difficult to learn. All it takes is a little dedication, some sweat, and a lot of patience in order to be successful. For our example I will use HyperCard's programming language HyperTalk to demonstrate many important aspects of programming.